Bundesamt
für Sicherheit in der
Informationstechnik

BSI – Technical Guideline

Designation:	**Cryptographic Mechanisms: Recommendations and Key Lengths**
Abbreviation:	BSI TR-02102-1
Version:	2018-02
As of:	May 29, 2018

Version	Date	Changes
2016-01	14.12.2015	Removal of the chapter on SSH which was transferred to TR-02102-4. Withdrawal of the 224-bit key length for ECDH-based mechanisms as already announced in the past few years. Withdrawal of the hash functions with 112 bit security level. Inclusion of the SHA-3 family into suitable hash functions. Adjustment of the recommendations given on RSA key lengths: greater emphasis is given to the fact that a change is recommended in the medium to long term for mechanisms based on the RSA problem and on the Diffie-Hellman problem in finite fields to key lengths of at least 3000 bits. Minor adjustments with respect to the possibility of precomputation attacks against the DH problem in finite fields. Update of various references.
2017-01	3.1.2017	Major revision of the section on random number generation on Windows systems. Adjustment of the security level of the present Technical Guideline to 120 bit, as far as concerns the post-2022 prediction period. The recommended key sizes for RSA and mechanisms based on the discrete log problem in finite fields have been adjusted accordingly.
2018-01	15.12.2017	Fundamental revision of the section on prime number generation. Revision of the statement on SHA-1 due to the publication of a SHA-1 collision. Shortening of document history to cover only the last three years in order to save space.
2018-02	9.5.2018	Correction of a problem related to compiling the document from its LaTeXsource.

Federal Office for Information Security
P.O.B. 20 03 63, 53133 Bonn, Germany
Email: tr02102@bsi.bund.de
Internet: https://www.bsi.bund.de

Contents

List of Tables

Notations and glossary

$\mathbb{F}_n$ The field with n elements. It is also referred to as $\mathrm{GF}(n)$.

$\mathbb{Z}_n$ The ring of the residue classes modulo n in $\mathbb{Z}$.

φ $\varphi : \mathbb{Z} \to \mathbb{Z}$ is the Euler's totient function. It can be defined by $\varphi(n) := \mathrm{Card}(\mathbb{Z}_n^*)$.

R^* The unit group of the commutative ring R.

Card For a finite set M, $\mathrm{Card}(M)$ (also written as $|M|$) refers to the number of its elements.

Ceiling function The ceiling function $\lceil \cdot \rceil : \mathbb{R} \to \mathbb{Z}$ can be defined by $\lceil x \rceil := \min\{z \in \mathbb{Z} : z \geq x\}$.

Floor function The floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$ is defined by $\lfloor x \rfloor := \max\{z \in \mathbb{Z} : z \leq x\}$.

A

AES Block cipher with a block size of 128 bits, standardised by NIST in FIPS 197 [43]. Corresponding to the length of the keys used, a distinction is made between AES-128, AES-192 and AES-256. Apart from related-key attacks against AES-192 and AES-256, no attacks against AES are known which provide a significant advantage over generic attacks on block ciphers.

Asymmetric cryptography Generic term for cryptographic mechanisms in which some cryptographic operations (such as the encryption of a message or the verification of a signature) can be carried out by parties who do not know secret data.

Authenticated encryption Encryption schemes are referred to as *authenticated* if not only the confidentiality, but also the integrity of the data to be encrypted is protected.

Authentication Objective of securely identifying a person or machine. In the given context, this applies to persons or machines who or which are the source or destination of a communication connection and the authentication is performed by making use of a cryptographic secret.

Authenticity Authenticity of a message means that no changes to the message were made since the message has been generated and that no incorrect information about the sender of the message is employed by the recipient. Within the meaning of the terminology used in this Technical Guideline, the authenticity of a message is protected reliably by means of a data authentication scheme only if the access to the authentication keys used is protected reliably by means of an instance authentication method and old messages are prevented by cryptographic mechanisms from being repeated.

B

Block cipher Key-dependent, efficiently computable, reversible mapping which maps plaintexts of a fixed given bit length n to ciphertexts of the same length. Without knowing the key, it should be practically infeasible to distinguish the output of the block cipher from the output of a randomly chosen bijective mapping.

C

Chosen-ciphertext attack Cryptographic attack in which the adversary can gain access to plaintexts corresponding to ciphertexts chosen by them. The aim of the adversary is usually to decipher a given ciphertext which does not belong to one of these plaintext–ciphertext compromises. Depending on whether the adversary knows this ciphertext before or after the end of the attack, a distinction is made between adaptive and non-adaptive chosen-ciphertext attacks.

Chosen-plaintext attack Cryptographic attack in which the adversary can gain access to ciphertexts for plaintexts chosen by them.

Collision resistance A function $h : M \to N$ is referred to as collision-resistant if it is practically infeasible to find $x \neq y$ with $h(x) = h(y)$.

Confidentiality Objective of binding read access to an information to the right to access. In a cryptographic context, this usually means that only the holders of a secret cryptographic key should be able to access the content of a message.

D

Data authentication Protection of the integrity of a message by means of cryptographic mechanisms.

Diffie-Hellman problem (DH) g, g^a, g^b are given, with g being a generator of the cyclic group G. g^{ab} must be calculated. The difficulty of this problem depends on the representation of the group. The DH problem can be solved easily by adversaries who are able to calculate discrete logarithms in G.

Discrete Logarithm (DL) Problem of calculating d given g^d in a cyclic group G generated by g. The difficulty of this problem depends on the representation of the group.

Disk encryption The term "disk encryption" refers to the complete encryption of a data medium. The objective of a disk encryption is to ensure that no confidential information can be read from the encrypted system, at least when it is switched off.

DLIES Discrete Logarithm Integrated Encryption Scheme, hybrid authenticated encryption scheme based on DH in $\mathbb{F}_p^*$.

E

ECIES Elliptic Curve Integrated Encryption Scheme, hybrid authenticated encryption scheme based on DH in elliptic curves.

F

Fault attack Attack on a cryptographic system in which the adversary uses an incorrect execution of a cryptographic operation and/or causes it actively.

Forward secrecy (for cryptographic protocols) Security property of a cryptographic protocol, which says that the disclosure of cryptographic long-term secrets does not enable an adversary to compromise previous sessions of the protocol [37]. It must be noted that forward secrecy can only be reached in any protocol if a random number generator which ensures at least Enhanced Backward Secrecy according to [62] was used within the protocol when generating the ephemeral keys. If *future* sessions which have not been manipulated by an adversary are also to remain protected in the case of a compromise of all long-term secrets, a random number generator which also offers enhanced forward secrecy [62] must be used when generating the ephemeral keys.

Forward secrecy (for deterministic random number generators) In the context of deterministic random number generators, forward secrecy means that future output values of the random number generator cannot be predicted with more than a negligible advantage by adversaries who only know previous output values of the random number generator, but not its internal state and whose computing power is below a limit which is given by the security level of the deterministic random number generator [62].

Forward security The term *forward security* for encryption and signature algorithms is related to the concepts of *forward secrecy* for cryptographic protocols and deterministic random number generators (but not identical to either of the two concepts); see, for example, the review article [58].

G

GCM Galois/Counter Mode, a mode of operation for block ciphers which constructs an authenticated encryption scheme on the basis of the block cipher. The authentication of data that is not encrypted is also supported.

GHASH Key-dependent tag which is used in the authenticated block cipher mode of operation GCM. The hash of a message $M = M_0 M_1 \dots M_n$ (M_i 128-bit blocks) in GHASH is $\sum_{i=0}^{n} M_i H^{n-i+1}$, with $H \in GF(2^{128})$ being the hash key. It must be noted that GHASH alone cannot be used as a cryptographic hash function or as a message authentication code! Good security properties are provided only if GCM (or the authentication of GCM without encryption, i.e. the GMAC authentication code) is used as a whole.

GMAC Message authentication code which results from using the GCM without data to be encrypted.

H

Hash function A function $h : M \to N$ which can be computed efficiently and for which M is significantly greater than N. h is referred to as a *cryptographic* hash function if it is collision-resistant and resistant to the calculation of first and second preimages. In this Technical Guideline, the term *hash function* usually refers to a cryptographic hash function.

Hybrid encryption Encryption scheme which uses public-key cryptography to transport keys for a symmetric encryption method which, in turn, is used to encrypt the message.

I

Information-theoretic security A cryptographic mechanism is referred to as *information-theoretically secure* if any adversary fails because of a *lack of information* when attempting to break the system. In this case, the security objective is achieved *irrespective of the computing power available to the adversary* as long as the assumptions regarding the system information which can be accessed by the adversary are valid. There are information-theoretically secure mechanisms in many areas of cryptography, for example for the encryption of data (one-time pad), for the authentication of data (Wegman-Carter MAC) or in the field of secret sharing (for Shamir's secret sharing algorithm, see also Chapter 8). In mechanisms of this kind, there are usually *no* security guarantees if the prerequisites for the operation of the mechanisms are not precisely complied with.

Instance authentication Proof of the possession of a secret by a user or an information-processing system to another party.

Integrity Objective of binding write access to an information to the right to change the information. In a cryptographic context, this means that a message can only be changed without being noticed using a certain secret cryptographic key.

K

Key length For symmetric cryptographic mechanisms, the key length, which is also referred to as key size, is simply the bit length of the secret key used. For RSA (signature and encryption algorithms), the bit length of the RSA module n is referred to as key length. For schemes which are based on the Diffie-Hellman problem or discrete logarithms in $\mathbb{F}_p^*$ (DLIES, DH key exchange, DSA), the bit length of p is defined as the key length. For schemes which are based on the Diffie-Hellman problem or discrete logarithms in an elliptic curve C over the finite field $\mathbb{F}_n$ (ECIES, ECDH, ECDSA and variations), the bit length of n is the key length.

M

MAC Message authentication code, key-dependent cryptographic tag. Without knowing the key, it should be practically infeasible for an adversary to distinguish the MACs of nonrecurrent messages from random data. In this case, tags cannot be forged successfully by an adversary with a probability that is considerably above 2^{-t}, with t referring to the length of the authentication tags. Here, requirements on the length of t greatly depend on the application.

Min entropy The min entropy of a discrete random variable X (intuitively of a random experiment with a countable set of possible outcomes) is defined as $-\log_2(p)$, with p referring to the probability of the most probable value for X.

P

Partition encryption The term "partition encryption" refers to the complete encryption of a partition of a data medium. The mechanisms used are similar to that of the disk encryption.

Preimage resistance A function $h : M \to N$ is referred to as preimage-resistant if it is practically infeasible to find a $x \in M$ for a given $y \in N$ so that $h(x) = y$. It is referred

to as resistant to the calculation of *second* preimages if it is virtually impossible to calculate a $x' \neq x$ for a given x, y with $h(x) = y$ so that $h(x') = y$.

Public-key cryptography See asymmetric cryptography.

R

Related-key attack Attack on a cryptographic mechanism, in which the adversary may query encryptions and decryptions not only under the key K which was actually used, but also under a number of other keys unknown to the adversary, which are related to K in a manner that is known to the adversary. This model is very advantageous to the adversary. In some situations, related-key attacks can still be relevant in practice, for example in connection to the construction of a cryptographic hash function based on a block cipher.

S

Secret sharing The term "secret sharing" refers to mechanisms used to distribute secret data (for example, of a cryptographic key) to several storage media. The original secret can only be reconstructed by evaluating several shared secrets. For example, a secret-sharing scheme can provide that, from a total of n shared secrets, at least k have to be known in order to reconstruct the cryptographic key to be protected.

Security level (of cryptographic mechanisms) A cryptographic mechanism achieves a security level of n bits if costs which are equivalent to 2^n calculations of the encryption function of an efficient block cipher (e.g. AES) are tied to each attack against the mechanism which breaks the security objective of the mechanism with a high probability of success.

Shannon entropy The Shannon entropy of a discrete random variable X (intuitively of a random experiment with a countable set of possible outcomes) is defined as $-\sum_{x \in W} p_x \log_2(p_x)$, with W being the range of values of X and p_x the probability with which X has the value $x \in W$.

Side-channel attack Attack on a cryptographic system which makes use of the results of physical measurements at the system (for example, energy consumption, electromagnetic emanation, time consumption of an operation) in order to gain insight into sensitive data. Side-channel attacks are very relevant to the practical security of information-processing systems.

Symmetric cryptography Generic term for cryptographic mechanisms in which all parties involved must have predistributed shared secrets in order to be able perform the entire mechanism.

T

TDEA Triple DES.

U

Uniform distribution In the context of this Technical Guideline, *uniformly distributed* generation of a random number from a base set M always means that it is practically impossible to distinguish the generating process from an ideally random (i.e. a truly random, uniformly distributed, independent) drawing of elements from M.

V

Volume encryption See partition encryption.

1. Introduction

With this Technical Guideline, the Federal Office for Information Security (BSI) provides an assessment of the security and long-term orientation for selected cryptographic mechanisms. However, no claim to completeness is made, i.e. mechanisms which are not listed are not necessarily considered by the BSI to be insecure.

Conversely, it would be wrong to conclude that cryptographic systems which use as basic components only mechanisms that are recommended in this Technical Guideline would automatically be secure: The requirements of the specific application and the linking of different cryptographic and non-cryptographic mechanisms can mean that in a given specific case the recommendations given here cannot be implemented directly or that vulnerabilities occur.

Due to these considerations, it must be emphasised in particular that the recommendations given in this Technical Guideline do not anticipate decisions, such as in the course of governmental evaluation and approval processes.

This Technical Guideline rather addresses primarily, in a recommendatory manner, developers who plan to introduce new cryptographic systems as from 2018. Therefore, this document deliberately dispenses with the specification of cryptographic mechanisms which are still considered to be secure at the time of the publication of this document, but can no longer be recommended in the medium term, as they do not have vulnerabilities that can be exploited yet, but have at least theoretical vulnerabilities.

Various other documents issued by the BSI and Federal Network Agency (BNetzA) can also play an important role in the development of new cryptographic systems, such as [2, 3, 4, 17, 27, 28, 29]. For certain specific applications, the recommendations included in these documents are even binding in contrast to the recommendations given in this Technical Guideline. A discussion of the regulations covered (as of 2011) can be found in [49].

The following two sections describe first both the security objectives and the selection criteria for the recommended cryptographic mechanisms. Moreover, very generic remarks on the practical implementation of the recommended mechanisms are provided.

In Chapter 2 to 9, the recommended cryptographic mechanisms are listed for the following applications

1. Symmetric encryption,
2. Asymmetric encryption,
3. Cryptographic hash functions,
4. Data authentication,
5. Instance authentication,
6. Key agreement,
7. Secret sharing and
8. Random number generators.

The key lengths required and other constraints to be observed are specified in the respective sections.

Often, various cryptographic algorithms must be combined with each other in order to ensure that the mechanism used meets the security requirements. For example, it is often necessary not only to encrypt confidential data, but the recipient must also be sure by whom the data was sent and/or if it was manipulated during the transmission. The data to be transmitted must therefore also be authenticated by means of an adequate method.

Key agreement schemes are another example. In this respect, it is important to know with whom the key agreement is performed in order to eliminate so-called man-in-the-middle attacks and unknown key share attacks [15]. This is achieved by means of schemes that combine key agreement and instance authentication.

For this reason, corresponding schemes are given in Appendix A for these two use cases. These schemes are designed by combining the schemes listed in Chapter 2 to 9 and comply with the security level required in this Technical Guideline.

In addition to this, frequently used algorithms are recommended in Appendix B. These algorithms are needed, for example, for the generation of prime numbers and other system parameters for asymmetric schemes, for the key derivation for symmetric algorithms etc.

In Appendix C, recommendations are given for the use of selected cryptographic protocols. In the current version of this Technical Guideline, this only applies to the SRTP protocol, since recommendations for TLS, IPsec and SSH were transferred to TR-02102-2, TR-02102-3 and TR-02102-4 [23, 24, 25].

It is planned to review the recommendations given in this Technical Guideline on an annual basis and to adjust them if necessary.

1.1. Security objectives and selection criteria

The security of cryptographic mechanisms depends primarily on the strength of the underlying algorithms. For this reason, this Technical Guideline recommends only mechanisms which can be assessed accordingly based on the results available today from long-standing discussions and analysis. Other factors relevant to security are the specific implementations of the algorithms and the reliability of potential background systems, such as required public key infrastructures for the secure exchange of certificates. The realisation of specific implementations, however, is considered in this Technical Guideline just as little as any potential patent-related problems. **When choosing the mechanisms, care was taken to ensure that the algorithms are not subject to patents, but the BSI cannot guarantee this. This Technical Guideline also includes notes on potential problems occurring during the implementation of cryptographic mechanisms. These remarks, however, should not be considered an exhaustive list of such potential problems.**

Overall, all cryptographic mechanisms specified in this Technical Guideline achieve, with the parameters required in the specific sections, a security level of at least 100 bits. For the prediction period beyond the end of 2022, the use of mechanisms that achieve a security level of at least 120 bits is recommended.

The bit lengths recommended in this Technical Guideline for use in new cryptographic systems, however, go by this minimum level only to the extent that it is not fallen short of for any recommended mechanisms. The effective strength of the mechanisms recommended in this Technical Guideline is higher than 100 bits in many cases. Thus, a certain security margin with respect to possible future progress in cryptanalysis is achieved.

As has already been stated in the introduction, it is not true either that, by the reverse implication, mechanisms which are not mentioned in this Technical Guideline do not reach the security level required.

Table 1.1 shows the key lengths of selected algorithms and types of algorithms for which will just about reach, according to current understanding, the security levels of 100 and 120 bits respectively.

Symmetric schemes		Asymmetric schemes		
Ideal block cipher	Ideal MAC	RSA	DSA/DLIES	ECDSA/ECIES
100	100	1900	1900	200
120	120	2800	2800	240

Table 1.1.: Examples of key sizes reaching a security level of 100 and 120 bits respectively

The *recommended* key lengths of different types of cryptographic primitive are summarised in Table 1.2.

Table 1.2.: Recommended key lengths for different cryptographic mechanisms

Block cipher	MAC	RSA	DH F_p	DH (elliptic curve)	ECDSA
128	128	2000 [a]	2000 [a]	250	250

[a] For the period of use beyond 2022, the present Technical Guideline recommends to use a key length of 3000 bits in order to achieve a similar security level for all asymmetric schemes. The suitability of RSA, DSA and DLIES key sizes below 3000 bits will not be extended further. A key length of ≥ 3000 bits will be binding for cryptographic implementations which are to conform to this Technical Guideline as from 2023. Any key size ≥ 2000 is, however, in conformity with this Technical guideline until the end of 2022. More detailed information can be found in the Remarks 4 and 5 in Chapter 3.

In the case of message authentication codes (MACs), the length of the digest output is an important security parameter independently of the key length. Ideally, it should be impossible in practice for an adversary to distinguish a MAC from a random function of the same digest length. As long as this criterium is fulfilled, the adversary is reduced to forging messages by guessing verification tags, and their probability of success per trial is $2-n$, where n is the tag length. In many applications, $n = 64$ will be acceptable under these circumstances, i.e. a tag length of 64 bits.
In the case of block ciphers, the block width is also a security parameter which does not depend on the key length. In the absence of structural attacks on a block cipher, the main effect of a small block width is that keys have to be exchanged more frequently. The precise effects depend on the mode of operation used. In this Technical Guideline, block ciphers with a block width of less than 128 bits are not recommended.
Moreover, an important type of cryptographic primitives which do not process any secret data at all are *cryptographic hash functions*. Here, the length of the returned digest value is the most important security parameter and should be at least 200 bits for general applications in order to ensure that the minimum security level required in this Technical Guideline is reached. The hash functions recommended in this Technical Guideline have a minimum hash length of 256 bits. Deviations from this rule for special applications will be addressed in this Technical Guideline where appropriate.
Diffie-Hellman key exchange schemes are to be handled in Table 1.1 and Table 1.2 in accordance with DSA/ECDSA.

1.2. General remarks

Reliability of forecasts regarding the security of cryptographic mechanisms When defining the size of system parameters (such as key length, size of the image set for hash functions and the like), not only the best algorithms known today for breaking the corresponding

mechanisms and the performance of today's computers must be taken into account, but foremost a forecast of the future development of both aspects must be taken as a basis, see [63, 64].

Assuming that there will not be a cryptographically relevant application of quantum computers in the next ten years, the performance of computers can be predicted relatively well over a period of 10 years. Unfortunately, this does not apply to the scientific progress with respect to cryptanalytic methods. Any forecast exceeding a period of 6-7 years is difficult, especially in the case of asymmetric schemes, and even **for this period of 6-7 years, the forecasts can turn out to be wrong due to unpredictable developments.**

The information in this Technical Guideline is therefore only provided limited to a period of time up to the end of 2024.

General guidelines on the handling of confidential data with longer-term protection requirements Since an adversary can store data and decrypt it later, there is a general risk for the long-term protection of confidentiality. This results in the following direct consequences:

- The transmission and storage of confidential data should be limited to the extent necessary. This not only applies to plaintexts, but, for example, also especially to avoiding the storage of session keys on any type of non-volatile media and their undelayed secure deletion, as soon as they are no longer required.
- The cryptosystem must be designed in such a manner that a transition to longer key lengths and stronger cryptographic mechanisms is possible.
- For data the confidentiality of which is to be secured in the long term, it is recommended to choose for the encryption of the transmission via generally accessible channels, such as the Internet, the strongest possible mechanisms from the ones recommended in this Technical Guideline right from the beginning. In most contexts, AES-256, for example, must be considered to be stronger than AES-128 due its longer key length. Since such general estimates are difficult, however, – in this particular example, AES-192 and AES-256 are *weaker* against the best attacks known than AES-128 [13] in some (construed) scenarios – an expert should already be consulted at this point if possible.
- With respect to the selection of cryptographic components for a new application, it must generally be taken into account that the overall system will, as a general rule, not be stronger than the weakest component. If a security level of 128 bits, for example, is thus aimed at for the overall system, all components at least have to comply with this security level. Choosing individual components which reach a higher security level against the best attacks known than the overall system, can nevertheless make sense as described above, since this increases the robustness of the system against progress in cryptanalysis.
- In order to minimise the possibility of side-channel attacks and implementation errors, the use of open source libraries should be given preference to in-house development in software implementations of the cryptographic mechanisms presented in this document if it can be assumed that the used functions of the library were subjected to a broad public analysis. When assessing a cryptosystem, the trustworthiness of all system functions must be tested. In particular, this also includes dependencies of the solution on properties of the hardware used.

Focus of this document The security of the cryptographic mechanisms recommended in this document is assessed without taking the use case into consideration. For specific scenarios, other security requirements may arise. It is possible that these are requirements which the mechanisms recommended in this Technical Guideline do not satisfy. Examples of this are the encryption of storage devices, the encrypted storage and processing of data on systems operated by external

providers („cloud computing“ and/or „cloud storage“) or cryptographic applications on devices with extremely low computational resources („lightweight cryptography“). Remarks on some such situations can be found in Section 1.5.

This document can thus support the development of cryptographic infrastructures, but cannot replace the assessment of the overall system by a cryptologist or anticipate the results of such an assessment.

General recommendations on the development of cryptographic systems Below, several basic principles are to be described in key points, which are generally recommended to be observed when developing cryptographic systems.

- It is recommended to already seek the collaboration with experts in cryptography during the early planning stages of systems for which cryptographic components are required.
- The cryptographic mechanisms listed in this Technical Guideline must be implemented in trusted technical components in order to reach the required security level.
- Moreover, the implementations of the cryptographic mechanisms and protocols themselves must be taken into account in the security analysis in order to prevent, for example, side-channel attacks.
- The security of technical components and implementations must be demonstrated according to the applicable protection profile by means of Common Criteria certificates or similar BSI procedures, such as in the course of an approval process, if the compliance of a product with the requirements of this Technical Guideline is to be shown.
- After the cryptographic system has been developed, the system should be evaluated by experts who were not involved in the development before it is used in a production environment. An assessment of the security of the mechanism solely by the developers should not be considered to be reliable even if the developers of the system have a sound cryptographic knowledge.
- The consequences resulting from a failure of security mechanisms used should be documented thoroughly. Wherever possible, the system should be designed in such a way that the failure or manipulation of individual system components is detected immediately and the security objectives are maintained by means of a transition into an adequate secure state.

1.3. Cryptographic remarks

Often, a cryptographic mechanism can be used for multiple applications, for example signature algorithms for data authentication and instance authentication. In this case, different keys should generally be used for the respective different applications.

Another example is symmetric keys for encryption and symmetric data authentication. Here, it must be ensured in practical implementations that different keys which cannot be derived from each other are used for both schemes, see also Section A.1.

In some cases, this Technical Guideline provides only an informative description of the cryptographic primitives. The cryptographic security, however, can only be assessed within the framework of the respective precise specification and the respective protocol used. For this reason, the corresponding standards referred to in this document are to be observed.

Further specific remarks are provided, where necessary, in the corresponding sections.

1.4. Handling of legacy algorithms

There are algorithms against which no practical attacks are known and which are still widespread in some applications and thus have a certain significance, but are nevertheless generally no longer assessed as conforming to the state of the art for new systems. Below, we briefly address the most important examples.

1. Triple-DES (TDEA) with keying option 1 [80]: The main aspects speaking against the use of 3-Key-Triple-DES in new systems are the small block width of only 64 bits, the reduced security against generic attacks on block ciphers as compared to AES as well as various cryptographic properties suboptimal even disregarding these preconditions. The existence of related-key attacks against Triple-DES with a compute effort of $\approx 2^{56}$ Triple-DES calculations [60], for example, should be mentioned. Even without taking related-key attacks into consideration, Triple-DES with keying option 1 has cryptographic properties which, according to present knowledge, do not indicate practically usable vulnerabilities, but are still more negative than one would expect for an ideal block cipher with an effective key length of 112 bits [65]. All in all, it is recommended in this document not to use Triple-DES in new systems unless it is absolutely necessary for reasons of backwards compatibility with an existing infrastructure. In this case, too, a migration to AES in the foreseeable future should be prepared.

2. HMAC-MD5: The lack of collision resistance of MD5 is not yet directly a problem for MD5 used in the HMAC construction [8], since the HMAC construction only requires a very weak form of collision resistance of the hash function. However, it does in principle not appear advisable to use primitives which were completely broken in their original function in new cryptosystems. Systems which use MD5 for cryptographic purposes are therefore not in conformity to the present Technical Guideline.

3. HMAC-SHA1: SHA-1 is not a collision resistant hash function; while the generation of collisions for SHA-1 takes considerable effort, it is nonetheless doable in practice [93]. From a security-technical perspective, however, there is, according to present knowledge, no reason speaking against using it in constructions which do not require collision resistance (for example, as a basis for an HMAC or as a component of a pseudo-random number generator). It is recommended to use a hash function of the SHA-2 family or of the SHA-3 family also in these applications as a general security safeguard. In principle, using SHA-1 in the HMAC construction or in other cryptographic mechanisms with comparable cryptographic requirements for the hash function used (for example, within the framework of a pseudo-random number generator or as a part of the mask generation function in RSA-OAEP) is in conformity with this Technical Guideline until 2018.

Triple-DES with keying option 2 according to [80] shows overall significantly more serious vulnerabilities with respect to chosen plaintext and known plaintext attacks in the single-key setting than with keying option 1 [69, 70]. Even though no ultimately practical attacks against TDEA with keying option 2 are known, it is recommended in this document not to use this cipher not only in new systems, but also to migrate existing cryptographic mechanisms using Triple-DES with two keys to AES (or at least to keying option 1 according to [80]) as soon as possible.
To the extent that Triple-DES is still being used, all recommendations for its use from [80] must be taken into account.

1.5. Important topics not covered in this Technical Guideline

Without any claim to completeness, we list explicitly once again several important topics which are not covered in this Technical Guideline:

1. Lightweight cryptography: Here, particularly restrictive requirements are imposed on processor and storage demands of the cryptographic methods used. Depending on the application, the security requirements might also be different than is usually the case.

2. When using cryptographic mechanisms in areas in which narrow provisions regarding the response times of the system have to be met, there may also be special situations which are not covered in this Technical Guideline. The recommendations on the use of SRTP given in Appendix C cover parts of this topic.

3. Hard disk encryption: Here, the problem is that encryption with data expansion as well as a significant expansion of the amount of data which has to be read by the storage medium and/or written to the storage medium are not acceptable in most contexts. None of the encryption modes recommended is immediately suitable as a basis of an application for hard disk encryption. Provided that an adversary cannot combine images of the disk state at several different points in time with each other, XTS-AES offers relatively good security properties and good efficiency [74]. If the adversary is able to create copies of the encrypted storage medium at a larger number of different points in time, some not necessarily insignificant leakage of information must be expected. The adversary can, for example, by comparing two images of a hard disk encrypted with XTS-AES, that were made at different points in time, immediately recognise which plaintext blocks on the hard disk were changed within this period of time and which were not.

4. When encrypting a solid-state drive, it is important in this context that the SSD controller does not implement the overwriting of the logical storage addresses physically in-place, but distributed to different physical storage areas. Thus, the respective current state of an SSD always also contains information about certain earlier states of the storage medium. An adversary with sound knowledge of the functioning of the SSD controller could potentially take advantage of this in order to track successive states of a logical storage address. When an SSD is used, a single image of the encrypted storage medium is thus potentially more valuable to an adversary than a single image of a hard disk.

5. Problems similar to those occurring in the case of the encryption of data media arise in the case of the encrypted storage of entire logical drives on remote systems which are not under the control of the data owner („cloud storage“). If the provider of the remote server or their security safeguards are not trusted to a great extent, however, it must then be assumed in this situation that an adversary may prepare disk images at any point in time and without being noticed. If files containing sensitive data are being held on a storage system that is under third-party control, then strong file encryption should be applied before transmitting the data. This is true even if volume encryption is applied prior to transmission of the data to the storage medium. Using a volume encryption solution on its own is only recommended if it includes an effective cryptographic protection against the manipulation of the data and the other prerequisites for the use of the corresponding mechanism in general cryptographic contexts are complied with (for example, the requirement of unpredictable initialisation vectors).In particular, the choice of mechanism should ensure that, unlike in XTS mode, no significant leakage of information by the application of frequency analysis to successive states of the storage medium is to be expected if the same data block is repeatedly written to.

6. Side-channel attacks, physical security questions are only covered marginally. To the extent they are provided, statements on side-channel attacks in this Technical Guideline should only be seen as exemplary references to possible threats from this direction **without any claim to completeness**! This Technical Guideline mainly addresses only those security aspects of cryptographic systems which can be reduced to the algorithms used.

Physical aspects such as the emission security of information-processing systems or cryptographic systems the security of which is based on physical effects (for example, quantum-cryptographic systems) are not covered in this Technical Guideline.

7. None of the data encryption schemes and protocols described in this Technical Guideline achieves, by itself, the objective of *traffic flow confidentiality*. Traffic flow analysis, i.e. an analysis of an encrypted flow of data taking into account source, target, time the connection has been established, size of the data packets transmitted, data rate and time of the transmission of the data packets, can allow significant inferences as to the contents of encrypted transmissions, see for example [6, 32, 92]. Traffic flow confidentiality is an objective which usually can be achieved only with a great deal of effort and which will thus not be achieved in many applications processing sensitive information either. However, it should be reviewed in each individual case by experts *how much* and *which* confidential information in a given cryptosystem is disclosed by means of traffic flow analysis (and other side-channel attacks, of course). Depending on the particular situation, the outcome of such an examination can require significant changes to be made to the overall system. It is therefore recommended to take into consideration the resistance to the disclosure of sensitive information under traffic flow analysis in the development of new cryptographic systems as an objective right from the beginning.

8. The security of the end points of a cryptographically secured connection is essential for the security of the data transmitted. When developing a cryptographic system, it must be documented clearly which system components have to be trusted so that the security objectives aimed at are reached. These components have to be hardened against being compromised in a manner that is suitable for the respective usage context. Corresponding considerations must include the entire life cycle of the data to be protected as well as the entire life cycle of the cryptographic secrets generated by the system. Cryptographic mechanisms can reduce the number of components of an overall system the trustability of which has to be ensured in order to prevent data from leaking, but cannot solve the basic problem of endpoint security.

9. Quantum resistant cryptographic mechanisms are with the exception of Merkle signatures not covered in the present Technical Guideline, because these mechanisms are only just reaching standardisation. As the development of quantum computers is an active area of research, further progress in this area must be monitored. An overview of the current state of development may be found in the study [10].

Therefore, this Technical Guideline does not give any recommendations with respect to the implementation of mechanisms in these areas. It is recommended to involve experts from the corresponding fields right from the beginning when developing cryptographic systems in general, but in these areas in particular.

2. Symmetric encryption schemes

Symmetric encryption schemes are used to ensure the confidentiality of data which is exchanged, for example, by means of a public channel, such as the telephone or Internet. The authenticity and/or integrity of the data is thus not ensured. For integrity protection, see Chapter 5 and Section A.1.

In this context, it should be emphasised that even in cases in which the protection of the confidentiality of the data transmitted seems to be the dominating or even sole security objective at first glance, neglecting integrity-securing mechanisms can easily result in vulnerabilities in the cryptographic overall system, which make the system also prone to attacks on confidentiality. Vulnerabilities to some types of active side-channel attacks may arise in such a manner. For an example, see [96].

This chapter first addresses symmetric methods, i.e. methods, in which the encryption and decryption keys are identical (in contrast to asymmetric schemes in which the secret key practically cannot be calculated on the basis of the public key without additional information). For asymmetric encryption schemes, which are only used as key transport methods in practice, see Chapter 3.

2.1. Block ciphers

General recommendations A block cipher is an algorithm which encrypts plaintext with a fixed bit length (e. g. 128 bits) by means of a key to ciphertext with the same key length. This bit length is also referred to as *block size* of the cipher. For the encryption of plaintexts with another length, so-called modes of operation are used, see 2.1.1. For new applications, only block ciphers the block size of which is at least 128 bits should be used.

The following block ciphers are recommended for use in new cryptographic systems:

AES-128, AES-192, AES-256, see [43].

Table 2.1.: Recommended block ciphers

In version 1.0 of this Technical Guideline, the block ciphers Serpent and Twofish were also recommended. There are no negative findings regarding these block ciphers, but the security of Serpent und Twofish has been examined far less intensively than that of the AES since the end of the AES competition. This applies both to classical cryptanalytic attacks and to other security aspects, for example the side-channel resistance of specific implementations. For this reason, this version of this Technical Guideline refrains from recommending other block ciphers apart from the AES.

Related-key attacks and AES In related-key attacks, it is assumed that the adversary has access to encryptions or decryptions of known or chosen plaintexts or ciphertexts by means of different keys which are related to each other in a way known to the adversary (i.e. for example differ in precisely one bit position of the key). Certain attacks of this type on round-reduced versions of the AES-256 [14] and on unmodified versions of the AES-192 as well as AES-256

[13] are the only known cryptanalytic techniques so far towards which AES shows a significantly poorer behaviour than an ideal cipher with a corresponding key length and block size.

At this point in time, these insights regarding the security of AES under specific types of related-key attacks do not have an impact on the recommendations given in this Technical Guideline. Due to the technical prerequisites of related-key boomerang attacks, a related-key boomerang attack on AES-256 from [13] with compute effort and data complexity of $2^{99.5}$, in particular, is not viewed as a violation of the security level of 120 bits that is aimed at in the medium term by this Technical Guideline.

The best known attacks on AES which do not require related-keys achieve only a minor advantage over generic attacks [18].

2.1.1. Modes of operation

As has already been established in Section 2.1, a block cipher by itself only provides a mechanism for the encryption of plaintexts of a single fixed length. In order to encrypt plaintexts of another length, an encryption scheme for plaintexts of (virtually) any length must be constructed based on the block cipher by means of an adequate *mode of operation*. As a further effect of a cryptographically strong mode of operation, it must be mentioned that the resulting encryption scheme will be stronger in some respects than the underlying block cipher, for example if the mode of operation randomises the encryption process and thus makes it difficult to recognise the same plaintexts that were encrypted several times.

Various modes of operation for block ciphers can initially only handle plaintexts the length of which is a multiple of the block size. In this case, the last block of a given plaintext may still be too short and must be padded accordingly, see Section 2.1.3 for adequate schemes. Among the modes of operation recommended for block ciphers, however, only the CBC Mode needs a padding step.

The easiest way to encrypt a plaintext the length of which is a multiple of the block size is to encrypt each plaintext block with the same key (this mode of operation is referred to as Electronic Code Book (ECB)). This results, however, in same plaintext blocks being encrypted to the same ciphertext blocks. The ciphertext thus provides at least information on the structure of the plaintext and a reconstruction of parts of the plaintext by means of frequency analysis may become realistic if the entropy per block of plaintext is low. In order to avoid this, the nth cipher block should not only depend on the nth plaintext block and the key used, but also on an additional value, such as the $(n-1)$th ciphertext block or a counter.

The following modes of operation are adequate for the block ciphers listed under 2.1:

1. Galois/Counter Mode (GCM), see [76],
2. Cipher Block Chaining (CBC), see [73], and
3. Counter Mode (CTR), see [73].

Table 2.2.: Recommended modes of operation for block ciphers

Remark 1 The Galois/Counter Mode (GCM) also ensures cryptographically secure data authentication given that the tag length is sufficient. For the other two modes of operation, it is generally recommended to provide separate mechanisms for data authentication in the overall system.Ideally, no decryption or other processing should be performed for unauthenticated encrypted data. If unauthenticated encrypted data is decrypted and further processed, there are increased residual risks with regard to the exploitation of error oracles.

2.1.2. Conditions of use

The modes of operation listed under 2.1.1 require initialisation vectors. Furthermore, certain constraints for secure operation must be complied with. These conditions are summarised as follows below.

1. For GCM:
   - Initialisation vectors may not repeat themselves within a key change period.
     More precisely, two AES encryptions (i.e. applications of the underlying AES block cipher) with the same input values (key, message) must not be carried out in the entire mechanism.
     Any non-compliance with this condition results in a potentially complete loss of confidentiality for the plaintext blocks affected if the repetition occurs as part of the generation of the GCM key stream!
   - Moreover, GCM requires the generation of nonces for the integrated authentication mechanism. According to [76], a bit length of 96 bits is recommended for them. This Technical Guideline follows that recommendation, with particular reference to the results from [59][1]. These initialisation vectors must not repeat themselves within the lifetime of an authentication key. In [76], it is required that the probability of a repetition of the GHASH initialisation vectors under a given key should be $\leq 2^{-32}$. This results in key change interval of at most $\approx 2^{32}$ calls of GHASH *irrespective of the GMAC tag length* [76] if the IVs are generated in a non-deterministic manner. In case of a deterministic generation of the initialisation vectors, it must be demonstrated that a repetition of IVs is ruled out throughout the entire lifetime of a key. If the GHASH initialisation vectors repeat themselves, there is the threat of a complete failure of the authentication mechanism!
   - For general cryptographic applications, GCM with a length of the GCM tags of at least 96 bits should be used. For special applications, shorter tags can be used as well upon consultation with experts. In this case, the guidelines for the number of allowed calls of the authentication function with a shared key from [76] must be complied with strictly.
2. For CTR: The counter values must not repeat themselves with the same key. Non-compliance with this condition results in a virtually complete loss of confidentiality!
3. For CBC: Only unpredictable initialisation vectors are to be used, see also Section B.2.

For methods recommended for generating unpredictable initialisation vectors, see Section B.2.

For applications for which the requirements mentioned in this document for the properties of the initialisation vectors cannot be met, it is urgently recommended to involve an expert.

2.1.3. Padding schemes

As already explained in Section 2.1.1, the CBC Mode requires an additional padding step: When partitioning a plaintext to be encrypted, it may occur that the last plaintext block is smaller than the block size of the cipher used. Formatting realised by filling this last block in order to achieve the size required is also referred to as *padding*.

The following padding schemes are recommended:

[1] In [59], errors in the proofs of security for the Galois/Counter Mode accepted until then are referred to and a corrected analysis of the security of GCM is presented. In this corrected analysis, a nonce length of exactly 96 bits turned out to be advantageous.

1. ISO padding, see [57], padding scheme 2 and [73], Appendix A.
2. Padding according to [86], Section 6.3.
3. ESP padding, see [85] Section 2.4.

Table 2.3.: Recommended padding schemes for block ciphers

Remark 2 In CBC mode, care must be taken to ensure that an adversary cannot find out based on error messages or other side channels if the padding of an introduced data packet was correct [96]. More generally, no side-channel information showing if a given ciphertext corresponds to a valid plaintext or if its format is invalid must be available to an adversary when using encryption schemes which allow an adversary to make changes to the ciphertext which, in turn, result in controlled changes to the plaintext.

2.2. Stream ciphers

In the case of stream ciphers, a key and an initialisation vector are used in the generation of a key stream, i.e. a pseudorandom bit sequence, which is then bitwise added to the message to be encrypted.

At the moment, no dedicated stream ciphers are recommended. AES in counter mode, however, can, of course, be understood as a stream cipher.

If a stream cipher is used, it is urgently recommended to protect the integrity of the information transmitted by means of separate cryptographic mechanisms. In the absence of such mechanisms, an adversary can make bit-by-bit changes to the plaintext.

2.3. Side-channel attacks on symmetric schemes

In addition to the security of the schemes used against cryptanalysis, the security of the implementation against side-channel and fault attacks is of vital importance for the security of a cryptosystem. This also applies to symmetric encryption schemes. A detailed examination of this topic is beyond the scope of this Technical Guideline and the countermeasures to be taken also heavily depend on the individual case. Nevertheless, the following safeguards are recommended in this document:

- Where it is possible at reasonable expense, cryptographic operations should be performed in security-certified hardware components (for example on a suitable smart card) and the keys used should not leave these components.
- Attacks which can be carried out by remote, passive adversaries are naturally difficult to detect and can thus lead to a significant unnoticed leakage of data over a long period of time. This includes, for example, attacks exploiting variable bit rates, file lengths or variable response times of cryptographic systems. It is recommended to thoroughly analyse the effects of such side channels on system security when developing a new cryptographic system and to take the results of the analysis into consideration in the development process.
- On the protocol level, the occurrence of error oracles should be prevented. This can be ensured most effectively by means of protecting all ciphertexts by a MAC. The authenticity of the ciphertexts should be verified before performing any other cryptographic operations and there should be no further processing of inauthentic ciphertexts.

In general, the generic recommendation of, wherever possible, using components which have already been subjected to an intensive analysis by a broad public and involving relevant experts in the development of new cryptographic infrastructures at an early stage also applies to this case.

3. Asymmetric encryption schemes

Due to their low efficiency as compared to widely used symmetric schemes, asymmetric encryption schemes are used in practice mostly for the transmission of symmetric keys, see also Chapter 7. The message to be encrypted (i.e. the symmetric key) is encrypted with the public key of the recipient. The recipient can then reverse the encryption using the secret key associated to the public key. It has to be practically impossible to reconstruct the plaintext from the ciphertext without knowing the secret key. This implies in particular that the secret key practically cannot be derived from the public key. In order to safeguard the attribution of the public key to the owner of the corresponding secret key, a public key infrastructure is usually needed.

For the specification of asymmetric encryption schemes, the following algorithms are to be defined:

1. one algorithm for the generation of key pairs (including system parameters).

2. one algorithm for the encryption of data and one algorithm for the decryption of data.

Simplifying slightly, the most practically relevant asymmetric encryption schemes and signature algorithms are based either on the difficulty of the problem of calculating discrete logarithms in suitable representations of finite cyclic groups or on the difficulty of decomposing large integers into their prime factors. Occasionally, the question arises as to which of these two approaches is to be regarded as cryptographically more stable. This Technical Guideline regards the factorisation of large numbers, the RSA problem, the problem of calculating discrete logarithms in suitable fields $\mathbb{F}_p$ (p prim), the problem of calculating discrete logarithms in suitable elliptic curves and the corresponding Diffie-Hellman problems as well-studied, hard problems and, in this respect, there is no reason for preferring mechanisms on the basis of discrete logarithms over mechanisms based on factorisation or vice versa. For particularly high security levels, using EC algorithms will be advantageous for reasons of efficiency, see also Table 3.2.

In addition to this, we give recommendations for minimum key lengths.

Remark 3 For asymmetric schemes, there is usually a number of equivalent, practically relevant representations of private and public keys. The bit length of the keys on a storage medium can vary depending on the chosen representation of the keys. For the definition of the key length for the recommended asymmetric cryptographic mechanisms, the glossary is thus referred to (entry Key length).

The following Table 3.1 provides an overview of the recommended asymmetric schemes and the key lengths l in bit.

Table 3.1.: Recommended asymmetric encryption schemes as well as key lengths and normative references

Method	ECIES	DLIES	RSA
l	250	2000 [a]	2000 [a]
Reference	[1, 50, 66]	[1, 50]	[88]
More detailed information in	Section 3.3	Section 3.4	Section 3.5

[a] For a period of use beyond 2022, it is recommended to use RSA/DLIES keys of a length of 3000 bits in order to achieve a uniform security level in all recommended asymmetric encryption schemes. The key length of 2000 is expected to remain in conformance with the present Technical Guideline until 2022. For further details, see Remark 4 as well as 5.

Remark 4 For mechanisms based on the Diffie-Hellman problem/calculation of discrete logarithms in elliptic curves, these recommended key lengths have , as far as the time frame up to 2022 is concerned, a slightly greater security margin as compared to the minimal security objectives of this Technical Guideline than it is the case with RSA schemes and schemes based on discrete logarithms in finite fields. In short, this is due to the following reasons:

1. The parameter sets for EC algorithms are standardised. A given set of security parameters is therefore used by a large number of users for many different applications and is thus a particularly worthwhile target for attack.

2. For *generic* elliptic curves, the most efficient known way of solving random instances of the Diffie-Hellman problem is to calculate discrete logarithms using various versions of Pollard's rho algorithm. In many cases, however, curve parameters are used in EC algorithms (for example for reasons of efficiency), which have obvious, non-generic properties or their generation was not documented fully. It is possible particularly for those curves that special properties are found which make the calculation of discrete logarithms easier there than in the generic case.

3. The computation of discrete logarithms in elliptic curves can be parallelized almost perfectly. Parallelising the number field sieve (i.e., the factoring of large integers/the computation of discrete logarithms in finite fields) is more difficult, especially as regards the matrix step.

With the current provisions, there is only a small buffer remaining between the security level of about 125 bits achieved by the recommended ECC bit lengths and the security level of 120 bits which this Guideline aims to meet for 2023 and beyond. In certain applications which have particularly high demands on security or which need security guarantees substantially exceeding the prediction period of the present Technical Guideline, it may be reasonable to arrange for significantly higher key sizes in ECC-based mechanisms. One example for such a situation is for instance given by the provisions on the key sizes of the Country Signer CA in [29]. As the security of Elliptic Curve Cryptography depends crucially on the adverary's inability to use any of the mathematical structure present in a given elliptic curve to compute discrete logarithms on the curve more efficiently than is allowed by the Pollard-Rho method, it remains conceivable that the requirements of this Technical Guideline may be revised upwards as part of applying precautionary principles.

It is recommended as a general security safeguard to use curve parameters in EC algorithms which were generated verifiably at random, the construction of which was documented in a traceable manner, and the security of which was subjected to a thorough analysis. An example of such curve parameters are the Brainpool curves [38].

Remark 5 Based on currently known factoring methods and assuming that quantum computers will not be used in such attacks, there is no indication that RSA moduli of length 2000 bit will be factorable in the near term. However, the security margin of cryptographic mechanisms with an assumed security level of about 100 bits is not large any more once any progress in cryptanalysis is assumed in addition to advances in computing technology. Within the last few years, the use of RSA keys with higher security levels has become technically more feasible. Moreover, raising the security level aimed at in this Technical Guideline enables on the one hand a harmonisation of the security margins of the schemes here recommended, and on the other hand a convergence in terms of security objectives to corresponding current international regulations, e.g. the SOGIS crypto catalogue [91]. The use of 3000 bit keys is therefore being recommended as a fundamental security measure for RSA as well as for cryptographic schemes based on the finite field Diffie-Hellman problem as from early 2023 and is a binding precondition for conformity to the present Technical Guideline for any systems with a corresponding projected lifetime.

Remark 6 The asymmetric cryptographic functions recommended in this document require further subcomponents (such as hash functions, message authentication codes, random number generators, key derivation functions, block ciphers) which must meet the requirements of this Technical Guideline if the security level aimed at is to be reached. In relevant standards [50, 66], sometimes the use of cryptographic methods is recommended which are not recommended in this Technical Guideline, and indeed in some cases for reasons of security (e.g. Two-Key Triple-DES in [50]). In general, it is recommended to follow two principles when implementing a standard:

- As cryptographic subcomponents, only the mechanisms recommended in this Technical Guideline should be used.
- If this is not compatible with standards compliance, an expert must be involved. The final decisions taken with respect to the cryptographic subcomponents chosen must be documented thoroughly and should be justified in the documentation under security aspects.

When selecting the recommended asymmetric encryption schemes, it has been ensured that only probabilistic algorithms[1] are used. Here, a **new** random value is required every time a ciphertext is calculated. The requirements for these random values cannot always be met directly by generating uniformly distributed values of a fixed bit length. More detailed information on these values is provided, where necessary, in the sections regarding the respective schemes.

3.1. Preliminary remark on asymmetric key lengths

3.1.1. General preliminary remarks

The assessments of the security of cryptographic mechanisms and key lengths included in this Technical Guideline are only valid until 2024, as has already been mentioned in the introduction. This restriction of the validity of this Technical Guideline is of particular importance for asymmetric encryption schemes. Below, we briefly explain the reasons for this. Afterwards, the question as to how the key lengths specified can be derived is briefly addressed.

3.1.1.1. Security of asymmetric schemes

As far as the mechanisms addressed in this Technical Guideline are concerned, the security of asymmetric cryptographic mechanisms is based on the assumed difficulty of problems in algorithmic number theory. In the case of RSA, it is the problem of calculating eth rooths in $\mathbb{Z}_n$, where n is a sufficiently large number of unknown factorisation into two prime factors p, q

[1] The RSA algorithm itself is not probabilistic, but the padding scheme recommended for RSA is.

and $e > 2^{16}$ is coprime to $\varphi(n) = (p-1)(q-1)$. The security of DLIES and ECIES can (as far as concerns the asymmetric component) be reduced to the Diffie-Hellman problem in the groups used. Thus, there are reductions to natural problems which are generally considered to be difficult for all recommended mechanisms.

However, as compared to the situation with symmetric encryption schemes, the long-term security of which is, of course, also generally threatened by unforeseen scientific progress, the following aspects have to be emphasised:

- With respect to the factorisation problem for general composite numbers and the problem of calculating discrete logarithms in $\mathbb{F}_p^*$, there has been more progress relevant to practice since the introduction of asymmetric cryptographic mechanisms than in the cryptanalysis of the most thoroughly studied block ciphers.

- In symmetric ciphers, the threat of active attacks (especially chosen-plaintext and chosen-ciphertext attacks) can partially be fended off by means of an adequate key management, especially by securely deleting symmetric keys after their intended lifetime has expired. In addition, if a symmetric cryptographic mechanism shows the first signs of vulnerability regarding chosen-plaintext attacks or chosen-ciphertext attacks, it is possible to migrate to another mechanism. In the case of asymmetric cryptosystems, however, at least the public keys associated to the ciphertexts of interest will always be available to the adversary.

- Furthermore, all asymmetric encryption schemes recommended in this Technical Guideline would become insecure in case of significant progress in the development of quantum computers.

As compared to the situation with digital signature algorithms, an adversary can also save any ciphertexts they can access for decryption at any later point in time. The objective of securing the authenticity of a signed document, on the other hand, can also be ensured retroactively by generating a new signature in a timely manner as long as the evidential value of the old signature algorithm can be considered to be given at the time the new signature is generated. Conversely, on the legal side of the issue, it is also possible to no longer accept signatures with cryptographically broken mechanisms at the time the signature is verified if there was no signature renewal with a valid mechanism. In contrast, there are usually no retroactive measures available to protect the confidentiality of a plaintext for a given ciphertext.

3.1.1.2. Equivalent key lengths for asymmetric and symmetric cryptographic mechanisms

The recommendations of this Technical Guideline for the key lengths of asymmetric cryptographic mechanisms are based on calculations on equivalences of symmetric and asymmetric key lengths, which are based on the following basic assumptions:

- For mechanisms based on elliptic curves: It is assumed that there is no method that solves the Diffie-Hellman problem on the curve used considerably faster than calculating discrete logarithms on the same curve. Moreover, it is assumed that the calculation of discrete logarithms on the elliptic curve used is not possible with significantly less complexity (measured by the number of performed group operations) than for generic representations of the same cyclic group[2]. For a generic group G, a computational complexity for the calculation of discrete logarithms of $\approx \sqrt{|G|}$ group operations is assumed.

[2] Algorithms operating on the generic representation of a group have only black box access to group operations and elements. Intuitively, one may imagine an oracle that accepts encrypted group elements as input and which outputs the result of any group operations encryptedly.

$log_2(R)$	ECDLP	Factorisation / DLP in $\mathbb{F}_p^*$
60	120	700
70	140	1000
100	200	1900
128	256	3200
192	384	7900
256	512	15500

Table 3.2.: Approximate computing power R required (in multiples of the computing power needed for a simple cryptographic operation, e.g. one-time evaluation of a block cipher on a single block) for the calculation of discrete logarithms in elliptic curves (ECDLP) and/or the factorisation of general composite numbers of the specified bit lengths.

- For RSA and methods based on discrete logarithms in $\mathbb{F}_p^*$: It is assumed that attacks which are more efficient than the general number field sieve when the parameters are chosen as recommended in this Technical Guideline will not become known throughout the prediction period of this Technical Guideline. For RSA and schemes based on discrete logarithms in $\mathbb{F}_p^*$, the same key lengths are recommended. In the case of schemes based on discrete logarithms, it is assumed that there is no method available to solve the Diffie-Hellman problem in a subgroup $U \subset \mathbb{F}_p^*$ with ord(U) prime more efficiently than by calculating discrete logarithms in U.
- It is assumed that attacks will not be carried out using quantum computers.

These assumptions are pessimistic from the adversary's point of view, because they do not include any scope for structural progress in the cryptanalysis of asymmetric schemes. Progress which is incompatible with the assumptions above can be of a very specialised nature and, for example, relate to new insights regarding *a single* elliptic curve. Although a calculation with 2^{100} elementary operations is considered to remain impractical within the timeframe relevant to this Technical Guideline, all recommended key lengths are therefore above the minimum security level of 100 bits aimed at in this Technical Guideline.For 2023 and beyond, a security level of 120 bits is consistently being aimed for, although a certain security margin is retained also under this regime for the methods based on elliptic curves.
With respect to mechanisms the security of which is based on the difficulty of calculating discrete logarithms, especially discrete logarithms on elliptic curves, attacks requiring oracle access to operations with the private key of a user can also be relevant. Such attacks can considerably simplify the calculation of discrete logarithms in a group; see, for example, attacks using a static Diffie Hellman oracle from [22, 33].
For the assessment of runtimes, we follow [39], Chapter 6. In particular, we assume, as [39], that the factorisation of a 512-bit number of general form corresponds approximately to the computing power needed for 2^{50} DES operations. Using the methods mentioned there results in the equivalences shown in Table 3.2, without any security margins for progress with respect to factorisation techniques and/or techniques for the efficient calculation of discrete logarithms in the groups in question (see [39], Table 7.2). For recommended key lengths, see Table 3.1.

3.1.2. Key lengths for information worthy of protection for a long period of time and in systems with a long planned period of use

For the purposes of this Section, *information worthy of protection for a long period of time* refers to information the confidentiality of which is to be maintained considerably longer than

the period of time for which this Technical Guideline forecasts the suitability of cryptographic mechanisms, i.e. for a period of time that is well beyond 2024. A reliable forecast on the suitability of cryptographic mechanisms is then no longer possible across the entire life cycle of the system. For this situation, it is recommended to provide security mechanisms which significantly exceed the minimum requirements of this Technical Guideline by involving an expert. By way of example, various possibilities of minimizing risks are illustrated below:

- When developing new cryptographic systems with a long projected period of use, it is recommended to provide already at the development stage the possibility of a future operation with larger key lengths. A change of the mechanisms used that might become necessary in the future and/or the practicability of such changes of the mechanisms should, if a long period of use is planned, also be taken into account already in the development of the original system wherever possible.

- Higher asymmetric key lengths than those required in this Technical Guideline should already be used when the system is introduced. One obvious possibility is to aim at achieving a uniform security level of ≥ 128 bits for all system components. Remarks on the minimum asymmetric key lengths required for different security levels can be found in Table 3.2 in this case.

- Generally speaking, the amount of information required to be protected in the long term, which is transmitted via public networks, should be reduced to the absolutely necessary extent. This applies in particular to information which is transmitted after it has been encrypted by means of a hybrid or asymmetric cryptographic mechanism.

- In order to achieve a certain level of quantum computer security, asymmetric schemes can also be fortified using additional symmetric methods (using symmetric long-term keys). In this respect, there are for example the following possibilities:
  - Usually, asymmetric cryptography, as has already been mentioned, is only needed in order to exchange a shared secret between the parties involved in the communication, from which symmetric session keys can be derived. Here, it is possible to integrate a cryptographic long-term secret into the key derivation function. An adversary who is able to efficiently solve the mathematical problem on which the asymmetric scheme is based will fail in correctly deriving the session key in this case as long as they do not know the symmetric key used by the key derivation.
  - Furthermore, it is possible to symmetrically encrypt an asymmetric key exchange by means of a pre-shared secret.

  In this case, the problem of distributing the mentioned long-term keys must, of course, be solved.

- Another possibility of fending off future attacks by quantum computers is, of course, to apply asymmetric cryptographic mechanisms which are assumed to be resistant to quantum computer attacks. This Technical Guideline does not contain recommendations regarding the use of encryption schemes that are resistant to quantum computers, since it is not believed likely that quantum computers will be used against the recommended mechanisms within the prediction period covered by this Technical Guideline. In the long run, a prognosis is difficult because various basic technical questions are still open. On the other hand, the choice of security parameters for the mechanisms recommended in this Technical Guideline, assuming purely classical attacks, is also understood by the scientific community significantly better than is currently the case for schemes that are resistant to quantum computers.The use of hybrid schemes, which combine a classical and a quantum resistant public-key encryption scheme in a suitable manner, in principle opens up a way to join

the security guarantees of proven classical methods against classical cryptanalysis with the advantages of quantum resistant methods. It is imperative that an expert be consulted regarding the choice and implementation of suitable post-quantum mechanisms as well as regarding their adequate combination with classical schemes and the choice of appropriate key sizes.

The consideration of the long-term stability of security properties is relevant foremost in connection with encryption schemes, because with signatures, at least from a technical point of view the possibility exists to generate a renewal signature. When planning the use of signature schemes in cryptographic infrastructures, it should therefore be ensured that a comprehensive implementation of signature renewal (and the general application of the steps necessary to properly verify documents protected by a renewed signature) will in fact be possible if it should become necessary to rotate some cryptographic scheme.

For a more detailed discussion regarding key lengths secure in the long term for asymmetric cryptographic mechanisms, we refer to [39, 64].

3.2. Other remarks

3.2.1. Side-channel attacks and fault attacks

In the context of asymmetric encryption schemes and/or asymmetric digital signature algorithms, various side-channel attacks can be relevant, the applicability of which must be reviewed for each given situation. This topic cannot be covered comprehensively in this Technical Guideline. The security of the implementation against side-channel attacks should be reviewed if there are relevant threat scenarios. The same applies to fault attacks.
More detailed recommendations on this topic can be found in [4] for cryptographic mechanisms on the basis of elliptic curves. A corresponding document for RSA, $\mathbb{F}_p$-DH and the corresponding signature algorithms is being prepared.

Of course, side-channel attacks also affect symmetric primitives, see Section 2.3.

3.2.2. Public key infrastructures

The asymmetric encryption schemes described in this Technical Guideline alone do not provide any protection against man-in-the-middle attacks. The security guarantees of the mechanisms described are only valid if man-in-the-middle attacks can be prevented reliably by additional methods.

Such attacks can only be fended off reliably if an authentic distribution of the public keys of all parties involved is ensured. For this purpose, there are various options available. However, generally, a public key infrastructure (PKI) is used. In a PKI, the problem of an authentic distribution of public keys is reduced to the distribution of the root certificates of the PKI.

When planning a PKI for an asymmetric encryption or signature system, it is recommended to take the aspects listed below into consideration. This list is not an exhaustive list of development requirements to be met by public key infrastructures, but only a list of relatively generic aspects the consideration of which seems reasonable when developing a PKI. Further requirements, which are not listed here, will usually arise when a system is developed and evaluated. The development of an adequate PKI for a new cryptographic application is not a trivial task and should be dealt with in close coordination with experts in these fields.

1. When issuing certificates, the PKI should verify if the applicant is in possession of a private key for their public key. This is possible, for example, by means of a challenge-response scheme for instance authentication requiring knowledge of the private key. It is also possible to generate the key pairs in an environment that is secure from the PKI's

perspective, if it is ensured that the generated key pairs are subsequently transported to the end user securely.

2. There should be ways to deactivate certificates in a timely manner and the adversary should not be able, without being noticed, to prevent information on the current status of a certificate from being available to the verifying party at the time of the verification.
3. Certificates should only be issued with a limited validity.
4. All issuers of certificates must be trusted.
5. A certificate should provide information as to whether authorisation for signing further certificates is granted. In general, each system that comes into contact with a certificate should be able to determine clearly what this certificate may be used for.
6. The length of certificate chains should be limited (by a value that is as low as possible).

3.3. ECIES encryption scheme

General description The abbreviation ECIES stands for *Elliptic Curve Integrated Encryption Scheme*. It is a hybrid encryption scheme. The security of the asymmetric component is based on the Diffie-Hellman problem in the respective elliptic curve used. Below, we describe a version of ECIES that is consistent with the other recommendations of this Technical Guideline. When describing the scheme, we closely follow [1].
The description of ECIES provided in this document is almost entirely identical to the description of the closely related DLIES scheme in Section 3.4. The main reason for addressing these two schemes separately is the difficulties which could result from differences in the notations as well as the recommendations regarding secure key lengths which differ for the two schemes. As a normative reference, ECIES-HC in [66] is recommended.
For an overview of the standardisation of ECIES and DLIES, we recommend [67].

Components ECIES requires the following components:

- A symmetric encryption scheme E_K. Here, all combinations of block ciphers and modes of operation recommended in this Technical Guideline are suitable.
- A Message Authentication Code MAC_{KM}. The schemes recommended in Section 5.3 can be used.
- A key derivation function H. H can simply be a hash function if its output has at least the length of the total symmetric key material to be derived. As an alternative, the key derivation function recommended in Section B.1 or one of the key derivation functions suggested in [66] can be used in order to generate key material of the desired length the given data.

Moreover, key material is needed as described in the section below on key generation.

Generation of keys

1. Generate cryptographically strong EC system parameters (p, a, b, P, q, i), see Section B.3.
2. Choose d randomly and uniformly distributed in $\{1, \ldots, q-1\}$.
3. Set $G := d \cdot P$.

Then, the EC system parameters (p, a, b, P, q, i), together with G, form the public key and d the secret key.

It is recommended to use the curve parameters specified in Table B.3.

Encryption Assume as given a message $M \in \{0,1\}^*$ and a public key (p, a, b, P, q, i, G) which can be attributed reliably to the authorised recipient R of the message. For encryption, the sender S then chooses a random number $k \in \{1, \ldots, q-1\}$ and calculates $B := k \cdot P$. S then calculates $X := k \cdot G$ and, based on this, $h := H(X)$. From h, sufficiently many bits are taken in order to create a key K for the symmetric encryption scheme as well as a key KM for the MAC. Based on the message M, a ciphertext $C := E_K(M)$ and a MAC $T := \mathrm{MAC}_{KM}(C)$ are then calculated. In the end, S sends the tuple (B, C, T) to R.

Decryption R receives (B, C, T) and calculates $X := d \cdot B$ as well as, based on this, $h := H(X)$, K and KM. R calculates $T' := \mathrm{MAC}_{KM}(C)$ and checks if $T = T'$. If this is not the case, the decryption process is aborted. If, however, $T = T'$, then R recovers the message by $M = E_K^{-1}(C)$.

Key length As concerns the order q of the base point P, at least $q \geq 250$ should apply.

A necessary prerequisite for the security of the ECIES scheme is the practical infeasibility of solving the Diffie-Hellman problem in the subgroup generated by P. According to current knowledge, this is the case for the recommended curve parameters.

Remark 7 Like DLIES, the scheme presented here is a probabilistic algorithm. Accordingly, a random value $k \in \{1, \ldots, q-1\}$ must be chosen at random from an approximately ideal distribution. See Section B.4 for an algorithm recommended for the calculation of the random value k.

3.4. DLIES encryption scheme

General description The abbreviation DLIES stands for *Discrete Logarithm Integrated Encryption Scheme*. It is a hybrid encryption scheme which, in the asymmetric component, is based on the difficulty of solving instances of the Diffie-Hellman problem in a suitable subgroup of $\mathbb{F}_p^*$. Below, we describe a version of DLIES that is consistent with the other recommendations of this Technical Guideline. When describing the scheme, we closely follow [1].

A normative description can be found in [50].

Components DLIES requires the following components:

- A symmetric encryption scheme E_K. Here, all combinations of block cipher and mode of operation recommended in this Technical Guideline are suitable.
- A Message Authentication Code MAC_{KM}.
- A key derivation function H. H can simply be a hash function if its output is at least equal in length to the entirety of the key material to be derived.

With respect to the recommended realisation of these components, the respective recommendations from Section 3.3 apply accordingly. Moreover, key material is needed as described in the following section on key generation.

Generation of keys

1. Choose randomly a prime q of suitable bit length (see the section on key size), such that q is prime.
2. Choose then k randomly with a bit length sufficient to ensure that kq is of the length of the key to be generated. Repeat until $p := kq + 1$ is prime.

3. Choose $x \in \mathbb{Z}_p^*$ such that $x^k \neq 1$. Set $g := x^k$. Then g is an element of order q in $\mathbb{Z}_p^*$.

4. Choose randomly a natural number a with $2 \leq a < q$ and set $A := g^a$.

Then (p, g, A, q) is the public key and a the secret key.

Encryption Assume as given a message $M \in \{0,1\}^*$ and a public key (p, g, A, q) which can be attributed reliably to the authorised recipient R of the message. For encryption, the sender S then chooses a random number $b \in \{1, \ldots, r-1\}$ and computes $B := g^b$. S then computes $X := A^b$ and, based on this, $h := H(X)$. From h, sufficiently many bits are taken in order to create a key K for the symmetric encryption scheme as well as a key KM for the MAC. Based on the message M, a ciphertext $C := E_K(M)$ as well as a MAC $T := \mathrm{MAC}_{KM}(C)$ are then calculated. In the end, S sends the tuple (B, C, T) to R.

Decryption R receives (B, C, T) and calculates $X := B^a$ as well as, based on this, $h := H(X)$, K and KM. R calculates $T' := \mathrm{MAC}_{KM}(C)$ and checks if $T = T'$. If this is not the case, the decryption process is aborted. If, however, $T = T'$, then R recovers the message by $M = E_K^{-1}(C)$.

Key length The length of the prime number p should be at least 2000 bits for a period of use until 2022, and thereafter at least 3000 bits. The length of the prime number q should in either case be at least 250 bits. Footnote a) for Table 3.1 and Remark 4 as well as Remark 5 from Chapter 3 apply accordingly.

A necessary prerequisite for the security of the DLIES scheme is the practical infeasibility of determining the discrete logarithm in the subgroup generated by g. According to current knowledge, this is the case with the recommended sizes of p and q. However, the difficulty of the problem of determining discrete logarithms in $\mathbb{F}_p^*$ can be reduced significantly by precomputations which only depend on p and not, for instance, on the chosen subgroup or their generator. As a general precaution, it is therefore recommended (but not strictly required for conformity with this Technical Guideline) to use key lengths of at least 3000 bits even before 2023 in cases in which a large number of users use a shared DH modulus instead of the minimum key length of 2000 bits required .

Remark 8 The DLIES scheme is a probabilistic algorithm, i.e. a random number k is required for the calculation of the ciphertext. Here, $k \in \{1, \ldots, r-1\}$ should be chosen with respect to the uniform distribution on $\{1, \ldots, r-1\}$. In Section B.4, three algorithms for the generation of k are discussed.

Remark 9 The efficiency of the scheme described at the beginning of this section can be improved if multiple users share the values (p, q, g), so that they can be precomputed. Alternatively it is also possible to use published parameter sets. The present Technical Guideline recommends in this case the use of the MODP groups from [84] or of the ffdhe groups from [11], in each case assuming a suitable choice of key sizes (this means that, for instance, MODP-1536 is *not* regarded as suitable, independently of the projected period of use). In the mentioned groups, one finds $q = (p-1)/2$ and has $g = 2$.
The use of a common p by multiple users is recommended only if $\log_2(p) \geq 3000$, since the computation of discrete logarithms can be simplified by precomputation attacks which depend only on p.

3.5. RSA

Generation of keys

1. Choose two prime numbers p and q at random and independently of each other. For further advice on prime generation see section B.5.

   Here, p and q should be of comparable bit length and not too close to each other: when for instance p and q are chosen independently of each other from too narrow an interval, attacks based on knowing the leading bits of p and q will apply. If p and q are chosen according to section B.5, no security problem appears here.

2. With the recommended key length of 2000 bits (see below), choose the public exponent $e \in \mathbb{N}$ under the constraints

$$\mathrm{ggT}(e, (p-1)\cdot(q-1)) = 1 \text{ and } 2^{16} + 1 \leq e \leq 2^{1824} - 1.$$

3. Calculate the secret exponent $d \in \mathbb{N}$ depending on e under the constraint

$$e \cdot d = 1 \bmod \mathrm{kgV}(p-1, q-1).$$

With $n = p \cdot q$ (the so-called modulus), (n, e) is then the public key and d the secret key. Moreover, the two prime numbers p and q must, of course, be kept secret, since otherwise anyone could calculate the secret exponent based on the public key (n, e) as described above in Item 3. It is recommended to not save any data from the generation of keys persistently apart from the keys generated and to overwrite all generated data in the computer memory after the keys have been generated. Furthermore, it is recommended to store private keys on a protected storage medium and/or in encrypted form in such a manner that only authorised users are able to perform decryption operations.

Remark 10 (*i*) The order in which the exponents are chosen, i.e. choosing e first and then d is intended to prevent small secret exponents from being chosen, see [20].

(*ii*) When using probabilistic primality tests to generate the two prime numbers p and q, the probability that one of the numbers is still composed should be at most 2^{-100}. See Section B.5 for suitable methods.

Encryption and decryption For encryption and decryption, see [88]. However, the message must be formatted to the bit length of the modulus n before the secret key d is applied. In this respect, the formatting method must be chosen carefully. The following scheme is recommended:

EME-OAEP, see [88].

Table 3.3.: Recommended formatting scheme for the RSA encryption algorithm

Usage of the older PKCS#1v1.5 paddings is not recommended, as in this context variations of the Bleichenbacher attack [16] have repeatedly turned out to be a problem, see e.g. [19] for a recent example.

Key length The length of the modulus n should be at least 2000 bits when the expected period of use extends at most to the end of 2022. Therafter, the present Technical Guideline requires a key size of at least 3000 bits. Foot note a) for Table 3.1 and Remarks 4 as well as 5 from Chapter 3 apply accordingly.

A necessary prerequisite for the security of the RSA scheme is the practical infeasibility of decomposing the modulus n into its prime factors without knowing p and q. According to current knowledge, this is the case for the recommended minimum bit length of 2000 bits.

4. Hash functions

Hash functions map a bit string $m \in \{0,1\}^*$ of any length[1] to a bit string $h \in \{0,1\}^n$ of a fixed length $n \in \mathbb{N}$. These functions play an important role in many cryptographic mechanisms, for example when deriving cryptographic keys or when authenticating data.

Hash functions $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ which are used in cryptographic mechanisms must meet the following three conditions depending on the application:

One-way property: For a given $h \in \{0,1\}^n$, it is practically impossible to find a value $m \in \{0,1\}^*$ with $H(m) = h$.

2nd preimage property: For a given $m \in \{0,1\}^*$, it is practically impossible to find a value $m' \in \{0,1\}^* \backslash \{m\}$ with $H(m) = H(m')$.

Collision resistance: It is practically impossible to find two values $m, m' \in \{0,1\}^*$ so that $m \neq m'$ and $H(m) = H(m')$ apply.

A hash function H which meets all of the conditions mentioned above is referred to as *cryptographically strong*.

Each of these three terms can be described more precisely in mathematical terms by comparing the best attacks known against these properties with optimal generic attacks.

The length of the hash output is a security parameter of crucial importance, because it determines the effort required for generic attacks. For the minimum security level of 120 bits required in this Technical Guideline, at least the condition $n \geq 240$ must be imposed on a hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ due to the birthday problem. It is not necessary here to distinguish different cases depending on the usage period of a system, as all hashing mechanisms recommended in this Technical Guideline already have a digest length ≥ 256 bits.

Remark 11 There are cryptographic applications of hash functions in which not all of the three mentioned properties of a strong hash function are required. On the other hand, there are further reasonable cryptographic requirements for hash functions which do not result from the three properties mentioned above. An example is the property of *Zero Finder Resistance* (resistance to the search for preimages of the hash value zero, [21]), which is important in connection with ECDSA signatures. The hash functions recommended in this Technical Guideline do not have any known cryptographic weaknesses that are of relevance to the cryptographic mechanisms recommended in this Technical Guideline, in which they are used.

According to present knowledge, the following hash functions are considered to be cryptographically strong and can thus be applied with all mechanisms used in this Technical Guideline:

- SHA-256, SHA-512/256, SHA-384 and SHA-512; see [41].
- SHA3-256, SHA3-384, SHA3-512; see [44].

Table 4.1.: Recommended hash functions

[1] Specifications of real hash functions usually include a length restriction which is so high that it is not exceeded by real input strings.

Remark 12 The hash function SHA-224 is no longer among the recommended algorithms. On the other hand, two families of hash functions are represented. The following remarks are in order:

1. Technically speaking, SHA-224 is in the context of this Technical Guideline a legacy mechanism. With a security level of approximately 112 bits, however, it has still to be considered as quite strong. The reason for its removal is that there are no advantages over the use of SHA-256 and SHA-224 fails to achieve the security level of 120 bits aimed at in this Technical Guideline for 2023 and beyond.

2. The hash functions of both the SHA-2 family and SHA-3 family are considered to be cryptographically strong. With respect to classical attacks on collision resistance and one-way properties, there is no practically relevant difference between the two function families that is known today. In certain other scenarios, there are differences; the functions of the SHA-3 family, for example, are resistant to length extension attacks.

Remark 13 (*i*) It is possible to generate colliding messages for SHA-1 with an effort of $\approx$ 10000 CPU years computation time [93]; the precise cost depends on the platform chosen to run the attack on and on the implementation of the attack. For further details, we refer the interested reader to [93].In addition, the effort needed generically to generate arbitrary SHA-1 collisions – which may be still relevant in cases where an adversary needs to achieve a great degree of control over the content of colliding messages – is $\approx 2^{80}$ SHA-1 evaluations and thus at the upper limit of potential practicality. In applications requiring a collision-resistant hash function, SHA-1 should definitively for this reason no longer be used.
(*ii*) Please note that a single collision of a hash function can already result in security problems in signature algorithms, see, for example, [36] and [47].

5. Data authentication

5.1. Preliminaries

In this Technical Guideline, data authentication refers to cryptographic mechanisms which ensure that transmitted or saved data cannot be changed by unauthorised persons. More precisely, the proving party (usually the sender of the data) uses a cryptographic key to calculate the tag of the data to be authenticated. The verifying party (usually the recipient of the data) then verifies if the received tag of the data to be authenticated corresponds to the tag which they would expect if the data is authentic and the correct key is used.

A distinction is made between symmetric and asymmetric schemes. In the case of symmetric schemes, the party giving proof and the verifying party use the same cryptographic key, which means that a third party cannot verify in this case who has calculated the tag or if it was calculated properly at all. In the case of asymmetric schemes, the private key is used for the calculation of the tag and verified with the associated public key.

In principle, the verifier of a message can therefore also forge messages when symmetric methods are used for data authentication. Such mechanisms are thus only suitable if the additional risk of compromise arising from the distribution of the symmetric key and its availability to (at least) two parties is acceptable. In addition, it must not be of concern if the verifying party forges a message. If one of these conditions is not met, symmetric data authentication methods are unsuitable and digital signatures must be used. In scenarios where these properties are unproblematic, the use of symmetric methods is more efficient. The integrity-protected transport of encrypted data over a network after negotiation of ephemeral keys is a standard case in which the use of symmetric methods for data authentication suggests itself.

5.2. Security objectives

When using cryptographic mechanisms for data authentication, a clarification of the security objectives to be achieved in the respective scenario is of crucial importance for the selection of mechanisms. Roughly speaking, for instance the following scenarios can be distinguished, which are important in many applications:

- Ensuring the integrity of data transmitted over a network on the way from the sender to the receiver. Here, sender and receiver usually have a common secret, and the receiver is not interested in producing forged transmissions. In this case, the use of a symmetric method for data authentication is therefore natural.

- Ensuring the non-repudiation of a message. This is to ensure that the owner of a particular key can be reliably identified as the originator of a message, and that even the author themselves can not create a signed message in such a way that doubts can subsequently be cast upon the validity of the signature. In such a situation, the verifiers of a message must not have the signature key. Therefore, in this case, only the use of digital signatures is possible. In addition to this, depending on the concrete scenario and the level of protection sought, the private signature key must also be protected from the signer themselves. This is, for example, the case if it is conceivable that the signer might subsequently invalidate past signatures by deliberately spreading their own private key. In addition, it must be ensured that the recipient is shown the message in the same way as the sender, and that

any unsigned portions (for example, the unsigned subject line in the case of a signed e-mail) can be unambiguously identified as such by the recipient and also by the sender.

- Protection of an asymmetric key exchange against man-in-the-middle attacks. In this case, no shared secret is available, and therefore an integrity-protected transmission of the key exchange messages must be ensured by means of digital signatures.

Remark 14 Special situations can lead to special requirements on the involved applications. For instance, code signatures have as security objectives the integrity of the transmitted application as well as the non-repudiation of possible malicious functionality within the deployed software, but the signed data can usually not be displayed in any meaningful way either to the recipient or to the originator, nor can a meaningful inspection of the signed content be performed with reasonable effort. The security functionality of a secure viewing component for the originator is therefore completely transferred to the originator's quality assurance processes and the security of the technical components they employ.

Remark 15 When processing authenticated data, only the data components that have actually been signed must be treated as having data integrity. It is not always trivial to enforce this principle, partly because the cases that may be critical to an application will never appear in legitimately signed data. In particular when using more complex signature formats (e.g. XML signatures) or when dealing with situations where security objectives which have not been foreseen by the developers of the components employed are to be enforced by the use of digital signatures, an expert should always carefully check whether additional safeguards may be necessary.

Remark 16 A signature may not yet be sufficient confirmation of the authenticity of signed data, because for instance replay attacks may be possible. Such attacks must be thwarted by additional measures. Generically, it is possible to achieve this by suitably combining data authentication schemes with instance authentication based on challenge-response methods. In some situations (e.g. software updates, key update) it can also be sufficient to check a version counter or a time stamp which is covered by the employed signature.

5.3. Message Authentication Code (MAC)

Message authentication codes are symmetric methods for data authentication which are usually based on block ciphers or hash functions. The proving party and the verifying party must have agreed upon a shared symmetric key beforehand. These mechanisms are usually used if large amounts of data are to be authenticated or if the verification or generation of tags have to be particularly efficient for other reasons. In many cases, both the confidentiality and the authenticity of the data must be ensured. See Section A.1 for those mechanisms. Moreover, see Chapter 7 for methods by means of which keys can be exchanged via insecure channels.

In general, the following schemes are considered to be secure if a block cipher listed in Table 2.1 is used in the CMAC scheme and in the GMAC scheme and/or a hash function listed in Table 4.1 is used in the HMAC scheme and the length of the key is at least 16 bytes for both schemes:

- CMAC, see [75],
- HMAC, see [8],
- GMAC, see [76].

Table 5.1.: Recommended MAC schemes

For the application of these schemes, the following recommendations must be observed:

1. As for the tag length, ≥ 96 bits are recommended for general cryptographic applications in all three schemes. As an absolute minimum for general applications, this Technical Guideline recommends 64 bits. Shorter tag lengths should only be used after all circumstances affecting the respective application have been taken into consideration by experts. For GMAC tags, there are attacks in which forgeries of tags of the length t for messages of a length of n blocks are possible with a probability of $2^{-t+\log_2(n)}$ per attempt and where this probability increases further if successful forgeries are detected [45]. This means that, with the same tag length, GMAC (and thus also the authenticated encryption mode GCM) provides a weaker protection of integrity than it is expected for CMAC or HMAC with the block ciphers and/or hash functions recommended in this Technical Guideline. The practical relevance of these attacks grows significantly if short authentication tags (< 64 bits) are used. The use of short tags with GMAC/GCM is therefore strongly discouraged.

2. The authentication keys used must be protected just as well as other cryptographic secrets in the same context.

3. In general, all requirements from [8, 75, 76] must be met for the scheme used respectively and their compliance documented.

With respect to the GMAC scheme, the other remarks regarding the usage conditions for GCM from Section 2.1.2 apply accordingly as far as the authentication function is concerned. The following table summarises the recommendations on key and tag length when MAC schemes are used:

Scheme	CMAC	HMAC	GMAC
Key length	≥ 128	≥ 128	≥ 128
Recommended tag length	≥ 96	≥ 96	≥ 96

Table 5.2.: Parameters for recommended MAC schemes

5.4. Signature algorithms

In signature algorithms, the data to be signed is hashed first and, based on this hash value, the tag and/or the signature is then calculated with the secret key of the prover. The verifier then checks the signature using the public key. As was already the case for asymmetric encryption schemes, it must not be possible in practice to calculate the signature without knowing the secret key. This implies in particular that the secret key cannot be practically derived from the public key.

For the distribution of the public keys to verifiers, a public key infrastructure is usually used. In any case, a reliable way of distributing the public keys (that is protected against manipulation)

is essential as is the case for all public key schemes.An in-depth discussion of the technical and organisational options to solve this problem would, however, exceed the scope of the present Technical Guideline, and the subject is therefore only treated on the margins.

For the specification of signature algorithms, the following algorithms must be defined:

1. An algorithm for the generation of key pairs.
2. A hash function which maps the data to be signed to a data block with a fixed bit length.
3. An algorithm for the signing of the hashed data and an algorithm for the verification of the signature.

In addition to this, we give recommendations for minimum key lengths.

For the calculation of the hash value, basically all hash functions listed in Table 4.1 are suitable. In the following four subsections, we only have to specify the algorithms and key lengths listed in Item 1. and 3. Furthermore, all recommended schemes can be used both for the signing of data and for the issuing of certificates.

Table 5.3 provides an overview of the signature algorithms recommended below.

1. RSA, see [53],
2. DSA, see [54] and [42],
3. DSA versions on elliptic curves:
   a) ECDSA, see [27],
   b) ECKDSA, ECGDSA, see [27, 54], and
4. Merkle signatures, see [31][a]

[a] Merkle signatures differ from the other signature algorithms recommended here in essential aspects. For a more detailed description of the most important aspects, see Section 5.4.4.

Table 5.3.: Recommended signature algorithms

Given a suitable choice of security parameters, all signature mechanisms here recommended reach a comparable level of security according to current knowledge if the private keys are reliably kept confidential and if in particular they cannot be determined by exploiting weaknesses of the implementation, for instance using side channel attacks, fault attacks or mathematical attacks aimed at weaknesses of key generation.

In the context of generating qualified electronic signatures under the scope of the Trust Services Act, formally different regulations may apply despite the fact that from a security technical point of view all recommended mechanisms are suitable. For further information, please refer to the SOGIS guidance on recommended cryptographic algorithms [91].

Remark 17 Apart from DS 3 (see Table 5.4), the asymmetric signature algorithms recommended are probabilistic algorithms[1]. Here, a **new** random value is required every time a signature is calculated. Requirements for these random values are specified in the corresponding sections.

[1] The RSA algorithm itself is deterministic, but not the padding schemes recommended for RSA, except for DS 3.

Remark 18 In contrast to all other signature algorithms mentioned in this document, Merkle signatures are considered to be secure against attacks using quantum computers [31]. Moreover, they are the only scheme among those mentioned here that is *forward secure* in the sense of [9], see also [58] for more information on the topic of Forward Security.

5.4.1. RSA

The security of this scheme is based on the assumed difficulty of calculating eth rooths in $\mathbb{Z}/(n)$, where n is an integer of unknown factorisation into two prime factors p, q and e is an exponent which is co-prime to $\varphi(N) = (p-1)(q-1)$.

Generation of keys Key generation is identical to that of the RSA encryption scheme. For details, see Section 3.5. The signature verification key is of the form (n, e) (n composite, e invertierbar mod $\varphi(n)$, $2^{16} < e < 2^{1824}$) and the signature key is $d := e^{-1} (\text{mod } \varphi(n))$.

Generation and verification of signatures For the generation and/or verification of signatures, see [53]. However, the hash value of the message must be padded to the bit length of the modulus n before the secret key d is used. The padding scheme must be chosen carefully, see [34] for instance. The following schemes are recommended:

1. EMSA-PSS, see [88].
2. Digital Signature Scheme (DS) 2 and 3, see [56].

Table 5.4.: Recommended padding schemes for the RSA signature algorithm

Key length The length of the modulus n should be at least 2000 bits (for a usage period up to 2022) and at least 3000 for use from 2023 onwards. Footnote a) for Table 3.1 and Remarks 4 as well as 5 from Chapter 3 apply accordingly.

5.4.2. Digital Signature Algorithm (DSA)

The security of this scheme is based on the assumed difficulty of the discrete logarithm problem in $\mathbb{F}_p^*$.

Key generation

1. Choose two prime numbers p and q so that the following applies:

$$q \text{ divides } p-1$$

.

2. Choose x in $\mathbb{F}_p^*$ and calculate $g := x^{(p-1)/q} \bmod p$.
3. If $g = 1$, go to 2.
4. Choose a number $a \in \{1, \ldots, q-1\}$ and set $A := g^a$.

Then, (p, q, g, A) is the public key and a the secret key.

Generation and verification of signatures For the generation and/or verification of signatures, see [54] and [42].
The generation and verification of signatures require a cryptographic hash function. One of the hash functions recommended in this Technical Guideline should be used. The length of the hash values should correspond to the bit length of q. If none of the hash functions recommended in Table 4.1 has a suitable hash length, the q leading bits of the hash output should be used. If the length L_H of the hash value is *shorter* than the bit length of q, the resulting signature algorithm will have a security level of (maximum) $L_H/2$ bits.

Key length For a usage period up to and including 2022, the length of the prime p should be at least 2000. For signatures which are meant to remain valid beyond the end of 2022 without further measures (e.g. signature renewal), a key length ≥ 3000 bits is recommended.

Remark 19 The DSA scheme is a so-called probabilistic algorithm, since a random number k is needed to calculate the signature. Here, $k \in \{1, \ldots, q-1\}$ should be chosen according to the uniform distribution on $\{1, \ldots, q-1\}$. Otherwise, there are attacks, see [82]. In Section B.4, two algorithms for the calculation of k are reviewed.

Remark 20 Regarding the generation of system parameters, see remark 9.

5.4.3. DSA versions based on elliptic curves

The security of these mechanisms is based on the assumed difficulty of the discrete logarithm problem on elliptic curves.

Key generation

1. Generate cryptographically strong EC system parameters (p, a, b, P, q, i), see Section B.3.
2. Choose d randomly and uniformly distributed in $\{1, \ldots, q-1\}$.
3. Set $G := d \cdot P$.

Then, the EC system parameters (p, a, b, P, q, i), together with G, form the public key and d the secret key.

Generation and verification of signatures The following algorithms are principally suitable:

1. ECDSA, see [27].
2. ECKDSA, ECGDSA, see [27, 54].

Table 5.5.: Recommended signature algorithms based on elliptic curves

For the generation and verification of signatures, a cryptographic hash function is required. In general, all hash functions recommended in this Technical Guideline are suitable. The length of the hash values should correspond to the bit length of q. The other remarks on how to select the hash function from Section 5.4.2 apply correspondingly.

Key length All signature algorithms listed in Table 5.5 ensure a security level of 100 bits if $q \geq 2^{200}$ holds for the order q of the base point P and if it is assumed that the calculation of discrete logarithms on the curves used is not possible more efficiently than with generic mechanisms. It is recommended to choose $q \geq 2^{250}$.

Remark 21 Like the DSA scheme, all signature algorithms recommended in this section are probabilistic algorithms. Here, too, a random value $k \in \{1, \ldots, q-1\}$ must be chosen according to the uniform distribution, since otherwise there are attacks. See [82]. In Section B.4, two methods for the calculation of k are presented.

5.4.4. Merkle signatures

In contrast to the signature algorithms described so far, the security of the algorithm described in [31] is only based on the cryptographic strength of a hash function and a pseudo-random function family. In particular, no assumptions on the absence of efficient algorithms for problems from algorithmic number theory such as the RSA problem or the calculation of discrete logarithms are needed. It is therefore generally assumed that Merkle signatures would, unlike all other signature algorithms recommended in this Technical Guideline, also remain secure against attacks using quantum computers.[2]

As hash functions, all hash functions recommended in Table 4.1 are suitable. The required pseudo-random function family can be constructed through the HMAC construction based on the hash function used.

For a detailed description of the scheme, see [31].

Due to the generally low complexity theoretical assumptions on which the security of Merkle signatures is based, Merkle signatures seem to be a good scheme for the generation of long-term secure signatures. This also applies if it is assumed that attacks by quantum computers are not used during the period of time in which the signature is to remain valid.

Unlike in the case of the other signature algorithms described in this Technical Guideline, however, only a finite number of messages can be authenticated when Merkle signatures with a given public key are used. Furthermore, the compute effort for generating the public key is proportional to this number of messages to be authenticated und thus relatively high if a large number of messages is to be signed without the intermediate generation and authenticated distribution of a new public key. Results of practical experiments regarding the efficiency of all substeps (creation of signatures, generation of signatures, verification of signatures) of the scheme described in [31] and regarding the resulting key lengths and signature sizes can be found in Section 6 in [31].

5.4.5. Long-term preservation of evidentiary value for digital signatures

If the time during which the authenticity and integrity of the data to be protected by means of a data authentication system is to remain secure exceeds the prediction period of this Technical Guideline considerably, then irrespective of the recommendations given for mechanisms and key lengths for digital signatures and irrespective of the corresponding recommendations for qualified electronic signatures in [17], it is recommended to provide for the possibility of a future migration of the system to new signature algorithms or longer signature keys already at the development stage. This should include mechanisms for the signature renewal of old signed documents using the updated schemes.

More detailed information on this topic can be found in the Technical Guideline 03125 (TR-ESOR) [30].

It is explicitly *not* recommended to use MAC schemes for long-term data authentication, since the verifying party needs to have knowledge of the secret MAC key and the long-term risk

[2] A discussion of quantum security of the collision resistance of hash functions can be found in [12].

of a compromise of the secret key is therefore significantly higher than in the case of digital signatures.

6. Instance authentication

In this Technical Guideline, instance authentication refers to cryptographic protocols in which a prover demonstrates to a verifier that they are in possession of a secret. In the case of symmetric schemes, this is a symmetric key which has to be exchanged in advance. In asymmetric schemes, the prover shows that they are in the possession of a secret key. In this case, a PKI is usually required so that the verifier is able to attribute the associated public key to the prover. Password-based schemes are primarily used to activate chip cards or other cryptographic components. Here, the owner of the component demonstrates that they are in the possession of a password or a PIN. In this document, a PIN (*Personal Identification Number*) is simply considered to be a password consisting of the numbers 0-9.

Authentication should, where reasonable and possible, be reciprocal and may be accompanied by a key agreement in order to ensure the confidentiality and integrity of subsequent communications. See Chapter 7 for recommended key exchange and key agreement schemes and Section A.2 for recommended protocols which combine the two schemes.

Therefore, this chapter includes for the first two schemes (Sections 6.1 and 6.2) only general ideas on instance authentication and recommendations only regarding the corresponding cryptographic primitives. For the cryptographic protocols required, Section A.2 is referred to. In particular, only recommendations for key lengths etc. are given there, too.

6.1. Symmetric schemes

To obtain evidence from the prover (P) to the verifier (V) that P is in possession of the secret symmetric key, V sends a random value r to P. In order to ensure that the scheme reaches the minimum security level aimed at in this Technical Guideline, r should have a min entropy of 100 bits. If a large number of authentications is performed with the same secret key, the probability that two of these challenge values will ever collide should be limited to $\leq 2^{-32}$. P then uses the shared secret key K to calculate an authentication code of r and returns it to V. Then, V verifies this tag. Such schemes are also referred to as *challenge-response methods*, see Table 6.1 for a schematic representation.

Prover (P)		**Verifier (V)**
		Choose random value r
	$\xleftarrow{r}$ (Challenge)	
Calculate authentication code c		
	$\xrightarrow{c}$ (Response)	
		Verify authentication code

Table 6.1.: Schematic representation of a challenge-response method for instance authentication

The calculation and verification of the authentication code depends on the selected scheme. In general, all encryption schemes recommended in Chapter 2 and all MAC schemes recommended in Section 5.3 can be used. For recommended bit lengths and constraints on the random values used, see Section A.2.

6.2. Asymmetric schemes

As was already the case in the section above, challenge-response protocols are also used for instance authentication using asymmetric schemes. Here, the prover uses their secret key to calculate a tag for a random value r sent by the verifier. The verifier then verifies the tag by using the corresponding public key. In general, all schemes recommended in Section 5.4 can be used for this purpose. For recommended bit lengths and constraints to be met by the random values used, see also Section A.2.

Remark 22 Even if the signature algorithms for data authentication recommended in Section 5.4 can also be used for instance authentication, it should be ensured that the keys used differ. This means that a key used for the generation of signatures should not be used for instance authentication. This must also be indicated in the corresponding certificates for the public keys.

6.3. Password-based methods

Passwords to activate the cryptographic keys made available on cryptographic components, such as signature cards, are short in most cases to ensure that the owner of the components can also remember the password. In many situations, the character set allowed is also restricted to the numbers 0-9. In order to still reach an adequate security level, the number of access attempts is usually limited.

6.3.1. Recommended password length for the access to cryptographic hardware components

The following constraints are recommended:

1. In general, it is recommended to use passwords with an entropy of at least $log_2(10^6)$ bits. This can be achieved, for example, by means of an ideally random assignment of six-digit PINs (see also [40], Section 4.3.3).
2. The number of consecutive unsuccessful attempts to gain access must be limited tightly. In the case of a password entropy of $log_2(10^6)$ bits, a restriction to three attempts is recommended.

Table 6.2.: Recommended password lengths and recommended number of attempts to gain access for the access protection of cryptographic components

Remark 23 If access passwords for cryptographic components are not at least generated approximately ideally randomly by a technical process, but set by the user, it is urgently recommended to raise the user's awareness with respect to the selection of secure passwords. It is recommended to refrain from using purely numerical passwords (PINs) in this case. For passwords which are created using an alphabet which at least includes the letters A-Z, a-z and 0-9, a length of eight characters is recommended. Furthermore, it is recommended to take safeguards which exclude passwords that are easy to guess (for example, individual words in the respective national language or an important foreign language as well as dates in formats that are easy to guess). For an assessment of the security level of user-generated PINs and passwords, we refer to [79], Table A.1.

Remark 24 In some applications, using passwords with a lower entropy than recommended above may also be compatible with this Technical Guideline after all circumstances have been taken into consideration by experts. A single unauthorised attempt to gain access, however, should never be successful with a probability of success of more than $\approx 10^{-4}$. The number of consecutive unsuccessful attempts to gain access must be restricted tightly. The precise restrictions depend on the application. The residual risks should be documented thoroughly. It is recommended to inform the authorised user, in situations in which this is applicable, about any unauthorised attempts to gain access even if the component was not blocked subsequently.

Remark 25 (*i*) In order to prevent denial-of-service attacks or the deliberate blocking of the component, a mechanism to unblock the blocking must be provided. The entropy of the *personal unblocking key* (abbreviated PUK) should be at least 100 bits if offline attacks are possible.

(*ii*) If no offline attacks on the PUK are possible, it is recommended to use a PUK with a min entropy of 32 bits (for example, 10 digits) and to irrevocably delete the cryptographic secrets contained in the component after a relatively low number of attempts to gain access (for example, 20).

(*iii*) The general recommendation given above, which has an entropy of at least approximately 20 bits for the password used in a password-based authentication scheme, applies, of course, only to the authentication to a security component which does not allow offline attacks and which can reliably implement the restrictions stated regarding the number of permissible attempts to gain access. In other situations, in which these conditions are not met (for example, if a cryptographic secret that provides access to sensitive information is derived directly from the password), it is recommended to choose passwords by means of a method which offers an entropy of at least 100 bits. For access to data or for the authentication of transactions with high protection requirements, using one-factor authentication is generally advised against. In this situation, two-factor authentication by means of knowledge (of a password) and ownership (of a secure hardware component) is recommended.

6.3.2. Recommended methods for password-based authentication to cryptographic hardware components

For contact-based chip cards, the method is very straightforward. The password is entered on the PIN pad of the card reader and transmitted to the chip card without any cryptographic protection. Although there are no cryptographic mechanisms on the part of the card reader, a certified card reader should be used in order to prevent attacks by manipulation of the card reader itself.

For contactless chip cards, the communication between the card reader and the chip card can be intercepted even from some distance. Here, the password to activate the chip card cannot simply be sent from the card reader to the chip card.

The following password-based method is recommended for the protection of access to contactless chip cards:

PACE: Password Authenticated Connection Establishment, see [26].

Table 6.3.: Recommended password-based method for the protection of access to contactless chip cards

The method recommended in Table 6.3 does not only demonstrate to the contactless chip card that the user is in possession of the correct password, but also carries out a key agreement method so that confidential and authenticated communications can be performed afterwards.

Remark 26 Here, too, the number of attempts must be restricted. It is recommended to block the chip card after three unsuccessful attempts. The other remarks from Section 6.3.1 apply accordingly.

7. Key agreement schemes, key transport schemes and key update

Key agreement schemes are used to exchange an encryption key via an insecure channel. It is absolutely crucial that these schemes are combined with instance authentication schemes. Otherwise, it is not possible to decide with which party the key agreement is carried out (data authentication alone is not sufficient here, since an adversary might have recorded communications which were performed in the past in order to exploit the data recorded for an attack). For this reason, as has already been the case in Chapter 6, we only give very general ideas for key agreement schemes and refer to Section A.2 for specific schemes, i.e. for key agreement schemes which also include instance authentication.

After a successful key agreement, both parties are in the possession of a shared secret. For methods recommended for the generation of symmetric keys based on this secret, see Section B.1. Basically, using a key derivation function is recommended there.

In some situations, it may make sense to integrate a predistributed secret into the key derivation function. Thus, a separation of different user groups, for example, can be achieved. It is also possible to establish an additional line of defence against attacks on the key agreement scheme in this way. With respect to a separation of different user groups, it may also make sense to integrate additional public data which is specific to both communication partners into the key derivation.

It is recommended to use only key agreement schemes in which both communication partners contribute equal shares to the new key. The entropy contributed by both sides should at least equal 100 bits. When choosing a key agreement scheme for a specific application, it should also be taken into account whether, in the chosen protocol, one side has greater control of the key material than the other side and whether such an asymmetry has security-related effects in the respective application.

In addition to key agreement schemes, key transport schemes are also of practical importance. Secret key data is generated by one party and transported secured to one or several recipients. In this case, the recipients do not have any control over the distributed session keys. The generating party can be a trusted third party or one of the parties involved in the communcation. In the latter case, it is recommended that all parties involved only use keys which they have generated themselves for the transmission of their own sensitive data.

This section will also cover key update schemes. In this case, two parties already share a shared secret and derive a new key on the basis of this secret at the end of a key change period. This can be achieved by deriving new session keys from a permanent master key or also through an update transformation generating a new key based on the current key and possibly some additional data.

Preliminary remark: asymmetric versus symmetric key agreement schemes With asymmetric key agreement schemes, security properties can be achieved which cannot be realised solely using symmetric cryptography. For example, both recommended asymmetric key agreement schemes have the property of *(perfect) forward secrecy*. This means that an adversary who knows all long-term secrets (if any) of the two parties involved in the communication[1]

[1] Primarily the long-term secrets which have to be used to the secure the connection against man-in-the-middle attacks.

can still not determine the key negotiated during an uncompromised execution of the protocol if they cannot efficiently solve the mathematical problem (the Diffie-Hellman problem in the schemes presented here) on which the asymmetric scheme used is based. In comparison, symmetric key agreement schemes can at most assure that an adversary who knows all long-term secrets of the two parties involved cannot determine the results of *previous* properly performed key agreements.[2]

7.1. Symmetric schemes

Key transport In general, all of the symmetric encryption schemes recommended in Chapter 2 can be used to transport session keys. It is recommended to combine an encryption scheme recommended in Chapter 2 with a MAC from Section 5.3 (in the encrypt-then-MAC mode) in order to achieve a manipulation-resistant transmission of the key material.

Key agreement Key agreement schemes, too, can be realised solely on the basis of symmetric schemes provided that the existence of a shared long-term secret can be assumed. Key establishment mechanism 5 from [51] is a suitable scheme. If an implicit confirmation of the key through the possession of the same session keys is not sufficient for the respective given cryptographic application, it is recommended to extend this protocol by a further step for the confirmation of the key. As key derivation function (KDF), the mechanism recommended in Section B.1 should be used.

Key update In some situations, it may be useful to exchange the keys used in a cryptographic system synchronously for all parties involved without any further communication taking place. In this case, key update mechanisms can be used. Below, we assume that the master key K_t of a cryptosystem is to be replaced at the time t by means of such a scheme. For general applications, we recommend the following mechanism:

1. $K_{t+1} := \mathrm{KDF}(s, \mathrm{label}, \mathrm{context}, L, K_t)$.
2. Here, KDF is a cryptographic key derivation function according to [78]. s is the salt value used in the extraction step. Label and context are included in the key expansion step according to [81] provided in [78]. Here, label is a string which indicates the function of the key to be derived and context includes information on the further protocol context. L refers to the length of the key K_{t+1} to be derived and is also included in the expansion step.

It is absolutely necessary to ensure that, in the case of a derivation of additional key material from K_t, derivation parameters other than those for the derivation of K_{t+1} according to the scheme described are used. It is recommended to enforce this using suitable label values. Moreover, it is recommended to also encode at least the cryptoperiod t in label or context. As an additional safeguard, it can make sense to use a new salt value for each key derivation. It is recommended to securely delete K_t immediately after the calculation of K_{t+1} as well as all interim results of the calculation. For further recommendations on the implementation of these schemes, [78, 81] is referred to.

7.2. Asymmetric schemes

In general, all asymmetric encryption schemes recommended in Chapter 3 can be used for the transport of new session keys.

[2] *Merkle puzzles* are exempted in this respect insofar as they constitute a key agreement scheme with public keys solely using symmetric primitives [68]. This scheme, however, is only of academic relevance.

The following methods are recommended as asymmetric key exchange schemes:

1. Diffie-Hellman, see [71],
2. EC Diffie-Hellman (ECKA-DH), see [27].

Table 7.1.: Recommended asymmetric key agreement schemes

One needs to specify:

1. An algorithm for the definition of the system parameters,
2. An algorithm for key agreement.

7.2.1. Diffie-Hellman

The security of this mechanism is based on the assumed difficulty of the Diffie-Hellman problem in groups $\mathbb{F}_p$, p a prime number.

System parameters

1. Randomly choose a prime number p.
2. Choose an element $g \in \mathbb{F}_p^*$ with $\mathrm{ord}(g)$ prime and $q := \mathrm{ord}(g) \geq 2^{250}$.

The triplet (p, g, q) must be exchanged **authentically** between the parties involved in the communication beforehand. In principle, the same system parameters may be used by many users. See remark 9 regarding the generation of suitable system parameters.

Key agreement

1. A chooses according to the uniform distribution a random value $x \in \{1, \ldots, q-1\}$ and sends $Q_A := g^x$ to B.
2. B chooses according to the uniform distribution a random value $y \in \{1, \ldots, q-1\}$ and sends $Q_B := g^y$ to A.
3. A calculates $(g^y)^x = g^{xy}$.
4. B calculates $(g^x)^y = g^{xy}$.

The key agreement, too, must be secured by means of strong authentication in order to prevent man-in-the-middle attacks. The negotiated secret is then g^{xy}. A mechanism for the subsequent key derivation from the shared secret is recommended in Section B.1.

Key length The length of p should at least be 2000 bits. For a period of use beyond the year 2022, the key length should be at least 3000 bits. Footnote b) for Table 3.1 and Remarks 4 as well as 5 from Chapter 3 apply accordingly.

Remarks on the implementation When implementing the Diffie-Hellman protocol, there are several common implementation errors. Some of these implementation problems are addressed in [83]. It is recommended to observe Section 7 of [83] in particular.

7.2.2. EC Diffie-Hellman

The security of this mechanism is based on the assumed difficulty of the Diffie-Hellman problem in elliptic curves.

System parameters Choose cryptographically strong EC system parameters (p, a, b, P, q, i), see Section B.3. We refer to the elliptic curve defined as C. The cyclic subgroup generated by P is referred to as $\mathcal{G}$.

The system parameters must be exchanged **authentically** between the parties involved in the communication beforehand.

Key agreement

1. A chooses uniformly distributed a random value $x \in \{1, \ldots, q-1\}$ and sends $Q_A := x \cdot P$ to B.
2. B chooses uniformly distributed a random value $y \in \{1, \ldots, q-1\}$ and sends $Q_B := y \cdot P$ to A.
3. A calculates $x \cdot Q_B = xy \cdot P$.
4. B calculates $y \cdot Q_A = xy \cdot P$.

The key agreement, too, must be secured by means of strong authentication. The negotiated secret is then $xy \cdot P$.

A mechanism for the subsequent key derivation from the shared secret is recommended in Section B.1.

Wherever possible, it is recommended to test during the execution of the key agreement on both sides whether the points Q_A and Q_B have been chosen according to the protocol requirements and to abort the protocol when the test result is negative. When the protocol mentioned above is executed correctly, $Q_A \in \mathcal{G}$, $Q_B \in \mathcal{G}$, $Q_A \neq \mathcal{O}$ and $Q_B \neq \mathcal{O}$ should hold. As part of the testing of $Q_A, Q_B \in \mathcal{G}$, it should also be tested explicitly whether $Q_A, Q_B \in C$.

Further remarks can be found in Section 4.3.2.1 of [27].

Key length The length of q should be at least 250 bits.

Remarks on implementation When implementing Diffie-Hellman key exchange, there are several common implementation errors. Some of these implementation problems are addressed in [83]. It is recommended to observe Section 7 of [83] in particular. The remarks in Section 4.3 of [27] and AIS46 [4] must also be taken into account.

8. Secret sharing

In many cases, cryptographic keys have to be stored over a long period of time. This requires in particular that copies of these keys have to be created in order to prevent the loss of the keys. The more copies are generated, however, the greater is the probability that the secret to be protected is compromised.

In this chapter, we therefore describe a method which allows dividing a secret, such as a cryptographic key K, into n shared secrets $K_1, \ldots, K_n$ so that any $t \leq n$ of these shared secrets are sufficient to reconstruct the secret, but $t-1$ shared secrets do not provide any information about K.

Another application of this scheme is to ensure a four-eye principle or, more generally, a t-of-n-eye principle in order to distribute, for example, the password for a cryptographic component in such a way to n different users that at least t users are required to reconstruct the password.

The secret-sharing algorithm presented in this Technical Guideline was developed by A. Shamir, see [90]. Below, we assume that the secret to be distributed is a key K of bit length r: $K = (k_0, \ldots, k_{r-1}) \in \{0,1\}^r$.

For computing secret shares to n user so that t users can reconstruct the secret K, proceed as follows:

1. Choose a prime number $p \geq \max(2^r, n+1)$ and set $a_0 := \sum_{i=0}^{r-1} k_i \cdot 2^i$.
2. Independently of each other, choose $t-1$ random values $a_1, \ldots, a_{t-1} \in \{0, 1, \ldots, p-1\}$. The values $a_0, a_1, \ldots, a_{t-1}$ then define a random polynomial
$$f(x) = \sum_{j=0}^{t-1} a_j x^j$$
via $\mathbb{F}_p$, for which $f(0) = a_0 = \sum_{i=0}^{r-1} k_i \cdot 2^i$ holds.
3. Calculate the values $K_i := f(i)$ for all $i \in \{1, \ldots, n\}$.

Table 8.1.: Calculation of the secret shares in Shamir's secret-sharing algorithm

The shared secrets K_i, together with i, are then transferred to the ith user.

Remark 27 The coefficients $a_0, \ldots, a_{t-1}$ of an unknown polynomial f of the degree $t-1$ can be found by means of the so-called *Lagrange interpolation formula* when given t points $(x_i, f(x_i))$ as follows:

$$f(x) = \sum_{i=1}^{t} f(x_i) \prod_{1 \leq j \leq t, i \neq j} \frac{x - x_j}{x_i - x_j}.$$

In this way, it is possible in particular to calculate $a_0 = f(0)$ (and thus K) given t points. This is the basis for the algorithm described above.

In order to reconstruct the secret K based on t shared secrets $K_{j_1}, \ldots, K_{j_t}$ (with pairwise different j_l), $a_0 = \sum_{i=0}^{r-1} k_i \cdot 2^i$ is calculated as follows (please note that the calculation in Table 8.1 and Table 8.2 is carried out in $\mathbb{F}_p$, i.e. modulo p):

1. Calculate the value $c_j = \prod_{1 \leq l \leq t, j_l \neq j} \frac{j_l}{j_l - j}$ for all $j \in \{j_1, \ldots, j_t\}$.
2. Calculate $K = \sum_{l=1}^{t} c_{j_l} K_{j_l}$.

Table 8.2.: Reassembly of the shared secret in Shamir's secret-sharing algorithm

Remark 28 The condition $p \geq \max(2^r, n+1)$ ensures that the secret can be represented as an element of $\mathbb{F}_p$ on the one hand and, on the other hand, that at least n independent shared secrets can be generated. The algorithm achieves information-theoretical security, which means that it is not possible even for an adversary with unlimited resources to reconstruct the distributed secret without finding out t shared secrets or a value that was derived from the knowledge of t shared secrets in a suitable manner.
The security of the scheme does thus not depend on other security parameters that go beyond the condition above. However, it must be ensured by means of organisational and technical safeguards that an adversary is not able to gain knowledge of t shared secrets. Any communication about the secret shares must therefore take place in encrypted and authenticated form to the extent that an adversary can physically record or manipulate this communication.

In addition, information-theoretic security is only achieved if the a_i are for all $i > 0$ chosen truly at random and according to the uniform distribution on $\mathbb{F}_p$. In order to achieve at least complexity theoretic security, the a_i should therefore be generated using a physical random number generator of the functionality classes PTG.2 or PTG.3 or a deterministic random number generator of functionality class DRG.3 or DRG.4. The values returned by the random number generator must be post-processed in such a way that they follow the uniform distribution on $\mathbb{F}_p$. Methods suitable to achieving this are given in section B.4.

9. Random number generators

A large number of cryptographic applications require random numbers, such as for the generation of cryptographic long-term keys, ephemeral keys and system parameters or for instance authentication. This applies to symmetric and asymmetric encryption schemes as well as to signature and authentication schemes and padding schemes.

In the generation of random numbers, the goal is usually to produce values uniformly distributed on $\{0,1\}^n$. This is, however, not always the application required in a given application. For special applications, Appendix B contains algorithms with which random values with desired properties (e.g. uniformly distributed on $\{0,\ldots,q-1\}$) can be calculated from the output values of a random number generator.

For most cryptographic applications, unpredictability and secrecy of the random numbers and/or the values derived from them are essential. Even if an adversary knows long subsequences of random numbers, this should not enable them to determine predecessors or successors. In general, unsuitable random number generators can crucially weaken otherwise strong cryptographic mechanisms. For this reason, suitable random number generators must be used for cryptographic applications.

In the German certification scheme, AIS 20 [2] (for deterministic random number generators) and AIS 31 [3] (for physical random number generators) are binding. The common mathematical-technical annex [62] is of vital importance. Reference [62] supersedes the previous documents [89] and [61].

The mathematical-technical annex [62] defines functionality classes for physical random number generators (PTG.1 – PTG.3), for deterministic random number generators (DRG.1 – DRG.4) and for non-physical non-deterministic random number generators (NTG.1). The old functionality classes K2, K3, K4, P1 and P2 from [89] and [61] were retained (under new names) for the most part (with partially higher requirements) and new functionality classes were added: hybrid deterministic random number generators (functionality class DRG.4), hybrid physical random number generators (functionality class PTG.3) and non-physical non-deterministic random number generators (functionality class NTG.1). Furthermore, [62] explains the mathematical background and illustrates the concepts with numerous examples.

Below, the recommendations for random number generators in general cryptographic applications, which are given in the individual sections of this chapter, are summarised in key points:

- When using a physical random number generator, it is generally recommended to use a PTG.3 generator. This applies in particular to the generation of ephemeral keys when generating digital signatures and when negotiating keys based on the Diffie-Hellman method. In contexts in which the use of a certified cryptographic component is necessary for the generation of random numbers, this recommendation applies, of course, only if components that are certified accordingly are available. In other contexts, a PTG.3 generator can usually be constructed through cryptographic post-processing of the output of a PTG.2 generator, where the post-processing is implemented in software and compatible with the requirements of the functionality class PTG.3.

- For certain specific cryptographic applications, PTG.2 generators can also be used. This is the case if the advantage resulting for an adversary from the existence of minor biases or dependencies in the distribution of the generated random values is demonstrably low, for example when generating symmetric session keys.

- In general, PTG.3 generators and DRG.4 generators have, compared to PTG.2 generators and DRG.3 generators, the advantage of an improved resistance to side-channel attacks and fault attacks. In the case of a PTG.3 generator, for example, the permanent inflow of large amounts of entropy into the internal state means that possible side-channel attacks directed at the cryptographic post-processing are made considerably more difficult, since an adversary can combine information on the internal state at two consecutive points in time t and $t+1$ only with great difficulty.
- When using a deterministic random number generator, it is recommended to use a DRG.3 generator or a DRG.4 generator. It is advisable to generate the seed from a physical random number generator of class PTG.2 or PTG.3. If such a random number generator is not available, using a non-physical, non-deterministic random number generator might also be considered. For details, see Section 9.3 and Section 9.5.
- These requirements for the min entropy of the seed of a deterministic random number generator, of course, increase accordingly if an overall security level of more than 100 bits is aimed at for a cryptographic system. In the general case, a min entropy of the RNG seed of n bits is required for a system security of n bits.
- Using PTG.2 generators or DRG.3 generators in security-critical functions (e.g. generation of keys, generation of nonces in DSA-like methods) is advised against if the planned active period of use extends beyond 2020. The active period of use of a random number generator refers to the period of time throughout which the random number generator is actually used to generate security-critical secrets. The reason for this is the generally higher sensitivity of such random number generators to side-channel attacks, fault attacks, and, in the case of PTG.2 generators, also to the exploitation of statistical biases in the output of the random number generator.
- In applications in which it is plausible that an adversary who could use minor biases in the distribution of the generated random numbers would also be able to exploit this retroactively, PTG.2 random number generators should wherever possible be replaced by random number generators of the functionality classes PTG.3 or DRG.4 if the currently generated random values can be expected to still be targets of attack after 2020. In these cases, thus also the passive period of use of a random number generator is relevant to current planning.
- Both for physical and for deterministic random number generators, a resistance to high attack potential should be shown in the respective application context.

9.1. Physical random number generators

Physical random number generators use dedicated hardware (usually an electronic circuit) or physical experiments to generate 'true' randomness, i.e. unpredictable random numbers. Usually, one uses unpredictable behaviour of simple electronic circuits, as may be caused by various forms of noise within the circuit. In the end, the entropy of the signal is generally due on a physical level to quantum effects or to the amplification of environmental influences within a chaotic system, where the influences cannot be controlled or separately measured. Even if an adversary knows parts of the generated sequence of random numbers and knows the random number generator well, including the physical environmental conditions at the time of random number generation, they should only have a negligible advantage (ideally none at all) over blindly guessing the random numbers.

In many cases, a deterministic post-processing of the 'raw noise data' (usually digitised noise signals) is required to eliminate any biases or dependencies.

If a physical random number generator is used, it is generally recommended to use a PTG.3 generator in accordance with AIS 31 (see [62], Chap. 4). This applies in particular to applications in which an adversary is at least in principle able to combine information about different random numbers. For certain specific applications, a class PTG.2 random number generator is sufficient. This is generally the case if the potential damage caused by minor deviations from assumed properties of uniform distribution and independence for the outputs of the random number generator can be assessed with reasonable certainty to be negligible in the given application. An example of such a situation is the generation of symmetric keys for a block cipher. Provided that an implementation of the random number generator is required in a certified cryptographic component, the recommendation of using a PTG.3 generator, of course, only applies to the extent that there are suitable certified components.

It is possible to construct a PTG.3 generator from of a PTG.2 generator by cryptographically post-processing the output of the PTG.2 generator in a suitable manner. This post-processing can usually be implemented in software. The precise requirements for the post-processing can be found in [62]. Roughly speaking, the post-processing must implement a DRG.3-compatible deterministic random number generator and at least as much new entropy must always be added by a class PTG.2 random number generator to the internal state of the random number generator as is requested by the cryptographic application.

Random numbers from PTG.2-conformant random number generators have a high entropy, but can also show certain biases and/or dependencies. The question as to whether a PTG.2 generator is sufficient in a specific application should be clarified with an expert.

Broadly speaking, PTG.2- and/or PTG.3-conformant random number generators must fulfil the following properties:

1. The statistical properties of the random numbers can be described sufficiently well by means of a stochastic model. On the basis of this stochastic model, the entropy of the random numbers can be reliably estimated.

2. The average increase in the entropy per random bit is above a given minimum limit (close to 1).

3. The digitised noise signals are subjected to statistical tests online, which are suitable to detect inacceptable statistical defects or deteriorations in the statistical properties within a reasonable period of time.

4. A total failure of the noise source is de facto identified immediately. Random numbers which were generated after a total failure of the noise source must not be output.

5. If a total failure of the noise source or inacceptable statistical defects of the random numbers are identified, this results in a noise alarm. A noise alarm is followed by a defined, appropriate response (e.g. shutting down the noise source).

6. (Only PTG.3-conformant random number generators) The (possibly supplementary) strong cryptographic post-processing ensures that the security level of a DRG.3-conformant deterministic random number generator is still assured even if a total failure of the noise source is not noticed.

Hybrid random number generators combine the security properties of deterministic and physical random number generators. In addition to a strong noise source, hybrid physical random number generators of the functionality class PTG.3 are equipped with strong cryptographic post-processing with memory. This is typically realised by post-processing cryptographically the random numbers of a PTG.2-conformant random number generator in an appropriate manner.

The development and security-critical assessment of physical random number generators require comprehensive experience in this field. **It is recommended to seek advice of experts in this field at an early stage as needed.**

9.2. Deterministic random number generators

Deterministic random number generators (also called pseudo-random number generators) can compute a pseudo-random bit sequence of practically any length from a random value of a fixed length, the so-called *seed*. For this purpose, the internal state of the pseudo-random number generator is initialised with the seed first. Publicly known parameters can also be included in the computation. In each step, the internal state is then renewed, and a random number (usually, a bit sequence of a fixed length) is derived from the internal state and output. Hybrid deterministic random number generators refresh the internal state from time to time with 'true' random values (reseed / seed update). In this respect, various seed update schemes can be used (for example, at regular intervals or upon the request of the application).

The internal state of a deterministic random number generator must be protected reliably against readout and manipulation.

If a deterministic random number generator is used, it is recommended to use a DRG.3- or DRG.4-conformant random number generator against the potential of attack HIGH in accordance with AIS 20 (see [62]). Roughly speaking, this means among other things:

1. It is practically impossible for an adversary to calculate predecessors or successors for a known random number sequence or to guess them with a significantly higher probability than would be possible without knowing this subsequence.

2. It is practically impossible for an adversary to calculate previously outputted random numbers based on the knowledge of an internal state or to guess them with a significantly higher probability than would be possible without knowing the internal state.

3. (Only DRG.4-conformant random number generators) Even if an adversary knows the current internal state, it is practically impossible for them to calculate random numbers which are generated after the next reseed / seed update or to guess them with a significantly higher probability than would be possible without knowing the internal state.[1] Also with respect to implementation attacks, DRG.4 generators have certain advantages over DRG.3-conformant random number generators.

9.3. Non-physical non-deterministic random number generators

For many cryptographic applications, such as in e-business or e-government, neither a physical nor a deterministic random number generator are available, since they generally run on computers without certified cryptographic hardware. Non-physical non-deterministic random number generators (NPTRNG) are usually used instead.

Like physical random number generators, non-physical non-deterministic random number generators also generate 'truly random' random numbers. However, they do not use dedicated hardware, but system resources (system time, RAM contents etc.) and/or user interaction (keyboard input, movement of the mouse etc.). Non-physical non-deterministic random number generators are usually used on computers which were not developed specifically for cryptographic applications, for example on all common types of PCs, laptops or smartphones. Like physical random number generators, they aim for information-theoretic security by producing enough entropy.

[1] The term "significantly higher probability" refers to a probability which is at least higher than the probability of guessing the truly random values that are generated for the seed update. For each seed update, a min entropy of at least 100 bits must be generated.

A typical approach is as follows: Long bit strings of 'random data' (or more precisely: non-deterministic data) are generated, with the entropy per bit being usually rather low. This bit string is mixed with an internal state. On the basis of the internal state, random numbers are calculated and output afterwards. The best known representative of non-physical non-deterministic random number generators is /dev/random. In the mathematical-technical annex [62], a functionality class for such random number generators (NTG.1) is defined. Roughly speaking, NTG.1 random number generators must reliably estimate the amount of entropy collected during operational use and the output data must have a Shannon entropy of > 0.997 bit per bit of output.

Broadly speaking, this means among other things

1. The entropy of the internal state is estimated. If a random number is output, the entropy counter is reduced accordingly.
2. Random numbers may only be output if the value of the entropy counter is high enough.
3. It is virtually impossible for an adversary to calculate previous random numbers based on the knowledge of the internal state and the random bit strings used previously for seed updates or to guess them with a significantly higher probability than would be possible without knowing the internal state and bit strings.

For NPTRNG, it is of vital importance that the entropy sources used by the random number generator cannot be manipulated by an adversary in terms of a reduction in the entropy or become predictable if the adversary is equipped with precise information on the execution environment. This prerequisite is not a matter of course even if an actually good NPTRNG is used. An example of a situation which is critical in this respect is the usage of virtualisation solutions [87]. In this case, the output of an NPTRNG can be fully predictable in extreme cases if the system is started twice from the same system image and all entropy sources of the virtual system are controlled by the host computer.
When it is planned to use an NPTRNG as the sole or most important random number generator for a system that is intended to be used for the processing of sensitive data, an expert should always be consulted.

9.4. Various aspects

Hybrid random number generators combine security properties of deterministic and physical random number generators. The security of a class DRG.4 hybrid deterministic random number generator is primarily based on the complexity of the deterministic part which belongs to class DRG.3. While the random number generator is used, new randomness is added again and again. This may be (for example) at regular intervals or upon the request of an application.

In addition to a strong noise source, hybrid physical random number generators of the class PTG.3 are equipped with strong cryptographic post-processing with memory. Compared to PTG.2-conformant random number generators, the functionality class PTG.3 also has the advantage that the random numbers have neither biases nor exploitable dependencies. Especially for applications in which a potential adversary is at least in principle able to combine information about many random numbers (e.g. ephemeral keys), a physical random number generator should belong to functionality class PTG.3.

The derivation of signature keys, ephemeral keys and prime numbers (for RSA) or the like from the generated random numbers should be carried out using suitable algorithms (for elliptic curves, see [4], Sections 5.2 and 5.5.1). Roughly speaking, as little information as possible about the derived values (to be kept secret) should be available to a potential adversary. Ideally, all values within the respective permissible range of values occur with the same probability and

different random numbers should at least have no relations which can be exploited in practice. Just like the signature algorithms, the generation of signature keys, ephemeral keys and prime numbers which are to be kept secret can also become the target of side-channel attacks ([47, 4] etc.). This aspect is addressed explicitly in [62].

The functionality classes from [89] and [61] that are relevant to security-critical applications were retained in [62] for the most part under new names and with partially higher requirements (P2 → PTG.2, K3 → DRG.2, K4 → DRG.3). Furthermore, new functionality classes were added , which may be considered for use in these contexts (PTG.3, DRG.4, NTG.1).

9.5. Seed generation for deterministic random number generators

For the initialisation of a deterministic random number generator, a seed with an sufficiently high entropy is required. The seed should be generated with a physical random number generator of the functionality classes PTG.2 or PTG.3. On PCs, a physical random number generator is usually not available or this random number generator was at least not subjected to a thorough manufacturer-independent certification. In this case, using a non-physical non-deterministic random number generator is recommended.

For this purpose, NTG.1-conformant random number generators are suitable (high potential for attack). At the moment, there are no NTG.1-certified random number generators. Therefore, we specify suitable methods for seed generation for the two most important PC operating systems below.

The use of the seed generation methods recommended in the following two subsections, however, can only be viewed as secure if the computer is under the full control of the user and no third-party components have direct access to the entire internal state of the computer, as may be the case, for example, when the entire operating system is run in a virtual environment. This means that the existence of viruses or Trojan horses on this computer can be ruled out. The users must be informed about these risks.

9.5.1. GNU/Linux

The following method is recommended for the seed generation when running the GNU/Linux operating system.

Reading of data from the /dev/random device file

Table 9.1.: Recommended method for seed generation under GNU/Linux

Remark 29 The randomness supplied by the device file /dev/random has so far only been reviewed by the BSI for certain kernel versions and found to be suitable if used within a PC-like system. The Linux-RNG is in this context deemed to be suitable for cryptographic applications if it fulfills the requirements of the functionality class NTG.1 according to [62]. Random number generators of this functionality class have to block if they do not receive sufficient entropy to generate the requested amount of random bits (see criterion NTG 1.6 in [62]). Cryptographically, this requirement is justified by noting that a guarantee on the amount of entropy contained in the seed value is needed when generating a seed for a pseudo random number generator. Since, on the other hand, not much randomness is required in seed generation, this requirement should furthermore normally not be problematic. However, /dev/random does not fulfil the NTG.1

criteria to the same extent for all versions of the Linux kernel. For more specific information regarding the cryptographic assessment of /dev/random in various versions of the Linux kernel, please refer to [98].

Remark 30 In principle, the recommended method should also work with Linux systems which have a relatively low number of entropy sources, for example embedded systems. Since /dev/random blocks if the internal entropy pool is used up, however, it is possible in this case that the generation of random numbers gets very slow.If, however, entropy sources are so scarce that the generation of for instance a 128 bit seed to be used in a deterministic random bit generator becomes difficult, then it must be assumed that using a non-blocking mechanism would be a highly dubious choice from a security perspective.

Remark 31 Using /dev/urandom may be problematic from a security point of view in some contexts [48]. Independently thereof, /dev/urandom does not fulfill the requirements of functionality class NTG.1 according to [62]. In practice, this means that it is not obvious how much entropy will in fact be contained in a key generated using /dev/urandom. By consequence, also the security level of an application relying on such key material ultimately not possible to determine. For this reason, it is recommended to read from /dev/random only a seed value of suitable bit length and to generate cryptographic key material by using a strong deterministic random bit generator seeded by this value. Ideally, this deterministic random number generator should regularly receive fresh entropy from /dev/random.

Remark 32 In principle, it is of course possible to use /dev/random not just for the seeding of a pseudo random generator, but also to directly generate cryptographic keying material.

9.5.2. Windows

In contrast to the GNU/Linux system, there is currently no function examined by the BSI for operating systems of the Windows family which ensures adequately high entropy. For the generation of secure seeds, several entropy sources should therefore be combined in an appropriate manner. For example, under Windows 10 the following method may be considered for generating a seed value of 128 bit entropy:

1. Read 128 Bit of random data from the Windows API function *CryptGenRandom* into a 128 bit buffer $S1$.

2. Get a bit string S_2 with at least 100 bits of entropy from a *different source*. There are various ways to do this, for instance:

   - Utilise the timing of successive keystrokes made by the user: if these can be time to a precision of a millisecond (which needs to be checked!), then it is possible to conservatively estimate per-keystroke entropy to about three bits. In order to assess the temporal resolution of the measured time intervals, the entire processing chain must be investigated for the entropy-limiting factors. For instance, it is possible that the accuracy of the internal clock gives one limit to resolution, the polling frequency of the key board another, and the interval at which the used system timers are being updated yet another. It is recommended to also measure the distribution of keystroke timings in practical tests and to examine it for any anomalies. The sequence of obtained timings can then be encoded in a binary string B. One then sets $S_2 := \mathrm{SHA256}(B)$ and erases the gathered keystroke timing data (and any other data gathered in the process) from working memory by zeroizing the relevant memory blocks.

- User-initiated events: The timings T_1, T_2, T_3, T_4, T_5 of five user-triggered events are recorded using the Windows API functions *QueryPerformanceCounter()* or *GetSystemTimePreciseAsFileTime()*. These timings are usually accurate to about a microsecond precision. Under preconditions which will be detailed in the sequel and which have to be checked for plausibility in each particular case, one may then assume that the bit string $T := T_1||T_2||T_3||T_4||T_5$ will contain around 100 bits of entropy from the point of view of an adversary. The preconditions just mentioned are as follows:

  a) Each T_i must not be guessable with precision more than one second, even if the adversary knows T_j for all $j \neq i$.

  b) In this situation, it must not be possible to constrain the value of T_i to less than 2^{20} possibilities by other considerations (for instance, regarding the polling frequency of the key board), if any interval of one second length is given which contains T_i.

  As in the previous example, one then sets $S_2 := \text{SHA256}(T)$ and erases T from memory.
  It is not always entirely easy to fulfill the requirements on the independence and unpredictability of user-initiated events. The problem here is that the point in time at which the software asks the user to initiate an event may be tightly predictable if the timing of an earlier event is known. Likewise, it is possible that the temporal delay between the user being prompted to supply input and the user initiating the event may be predicted with an accuracy exceeding one second. The plausibility of such entropy estimates should be studied in each particular case.

- In a similar manner, also mouse movements initiated by the user may be used for gathering entropy. It is not easy to precisely determine the amount of entropy that may be gathered from mouse movements, but it seems very implausible that for instance the sequence of pointer positions corresponding to 60 seconds of discretionary mouse movements can be losslessly compressed to less than 100 bit of data. One then defines S_2 again by a SHA-2 hash over the recorded mouse events.

3. In all of these cases, a seed value S for a suitable random number generator can then be derived by setting $S := \text{SHA256}(S_1||S_2)$.

Remark 33 There is nothing known to the BSI to indicate that in the above example a 128 bit value read from *CryptGenRandom()* will not already contain approximately 128 bits of entropy. It is for this reason that it is acceptable to gather only 100 bits of entropy from an additional source, if for the seed value of a pseudo random number generator a total 128 bits of entropy are desired. However, *CryptGenRandom()* has been less intensively studied by parties independent of the vendor than is the case for example for the random number generator integrated into the Linux kernel. As a basic precautionary measure, it is therefore advisable to combine randomness taken from *CryptGenRandom()* with the output of another entropy source.

Appendix A.

Application of cryptographic mechanisms

The mechanisms explained in the previous chapters often have to be combined with each other in order to ensure the protection of sensitive data. In particular, sensitive data to be transmitted should not only be encrypted, but also authenticated in order to ensure that any change is noticed by the recipient.

Moreover, a key agreement must always be combined with instance authentication and an authentication of all messages transmitted during the key agreement in order to ensure that both parties can be sure with whom they are communicating. Otherwise, the communication can be compromised by a so-called man-in-the-middle attack. Other types of attacks on the authenticity of the communication than the man-in-the-middle attack can, depending on the application, also threaten the security of an information-processing system without instance authentication or without data authentication (e.g. replay attacks).

In this chapter, we provide adequate mechanisms both for encryption with data authentication and for authenticated key agreement.

A.1. Encryption schemes with data authentication (secure messaging)

In general, all schemes recommended in Chapter 2 and/or Section 5.3 can be used when encryption and data authentication are combined.

However, the following two constraints must be complied with:

1. Authentication covers only the encrypted data and the associated non-confidential data, if any, that is to be transmitted in unencrypted form and subject to authentication.Other data which is sent within the same transmission is *not* authenticated.

2. Encryption and authentication keys should be different and it should not be possible to derive them from each other.

Remark 34 It is possible to derive encryption and authentication keys from a shared key. Recommended schemes are summarised in Section B.1.

Remark 35 For the authenticated transmission of encrypted data, it is recommended to use a MAC in encrypt-then-MAC mode.

Remark 36 If data is being transmitted encryptedly for which non-repudiation of the plain text is a security objective, the plain text should be digitally signed. In this case, one may first sign the plain text, then encrypt it, and finally protect the integrity of the ciphertext by means of a MAC against tampering in transit. Using a signature on the ciphertext may be reasonable if in addition the ciphertext is meant to be non-repudiable or if only the sender should be able to change the ciphertext. However, usually the signer cannot meaningfully examine the ciphertext before signing it.

A.2. Authenticated key agreement

As has already been explained in the introduction to this chapter, key agreement must always be combined with instance authentication. After some general preliminary remarks, we not only provide schemes which are based on purely symmetric algorithms, but also schemes which are based on purely asymmetric algorithms.

A.2.1. Preliminary remarks

Objectives The objective of a key exchange scheme with instance authentication is that the parties involved have a shared secret and that they are sure with whom they share it after the protocol has been executed.

For the derivation of symmetric keys for encryption and data authentication schemes from this secret, see Section B.1.

Requirements on the environment Symmetric schemes for an authenticated key exchange always assume the existence of predistributed secrets. In the case of asymmetric schemes, the existence of a public key infrastructure which can reliably bind keys to identities and authenticate the origin of a key by means of corresponding certificates is usually assumed. Moreover, it is assumed that the root certificates of the PKI have been made known to all parties involved via reliable channels and that all parties involved are able to properly check the validity of all relevant certificates at all times.

Remarks on the implementation In the specific implementation of the schemes presented, the following two conditions must be met.

1. The random values used for the authentication must differ with high probability every time the protocol is executed. This can be achieved, for example, by choosing a random value of length at least 100 bits with respect to the uniform distribution from $\{0,1\}^{100}$ each time.

2. The random values used for the key agreement must at least reach an entropy that corresponds to the desired key lengths of the key to be agreed upon[1]. In addition, each party involved in the key agreement should at least contribute a min entropy of 100 bits to the key to be negotiated.

A.2.2. Symmetric schemes

In principle, each scheme from Section 6.1 for instance authentication can be combined with each scheme from Section 7.1 for key exchange. The schemes must be combined with each other in such a way that the exchanged keys are actually authenticated, which means that man-in-the-middle attacks can be excluded.

The following scheme is recommended for this application:

Key establishment mechanism 5 from [51].

Table A.1.: Recommended symmetric methods for authenticated key agreement

Remark 37 As encryption methods, all authenticated encryption schemes recommended in this Technical Guideline can be used in key establishment mechanism 5 from [51] (see Section A.1).

[1] Here, it is assumed that only symmetric keys are negotiated.

A.2.3. Asymmetric schemes

As already described for symmetric schemes, each scheme from Section 6.2 for instance authentication can also in principle be combined with each scheme from Section 7.2 for key agreement.

In order to prevent mistakes in self-designed protocols, however, the schemes for key agreement with instance authentication on the basis of asymmetric schemes listed in Table A.2 are recommended.

All recommended schemes need as a precondition a mechanism for the authentic distribution of public keys. This mechanism must have the following properties:

- The public key generated by a user must be bound reliably to the user's identity.
- The associated private key should also be bound reliably to the identity of the user (a user should not be able to register a public key under their identity for which they cannot use the associated private key).

There are several ways to achieve this. The authentic distribution of keys can be achieved by means of a PKI. Compliance with the requirement that the owners of all certificates issued by the PKI should actually be the users of the associated private keys can be tested by the PKI before issuing the certificate by executing one of the protocols for instance authentication described in Section 6.2 with the applicant, using their public key.
If the PKI does not carry out such testing, it is recommended to supplement the schemes recommended below by key confirmation step, in which it is verified that both sides have determined the same shared secret K and in which this secret is bound to the identities of the two parties involved. For the confirmation of the key, the scheme described in [77], Section 5.6.2, is recommended. In the second recommended method (KAS2-bilateral-confirmation according to [77]), this step is already included.

1. Elliptic curve key agreement of ElGamal type (ECKA-EG), see [27].
2. Instance authentication with RSA and key agreement with RSA, see KAS2-bilateral-confirmation according to [77], Section 8.3.3.4.
3. MTI(A0), see [52], Annex C.6.

Table A.2.: Recommended asymmetric schemes for key agreement with instance authentication

Remark 38 In order to comply with this Technical Guideline, care must be taken in the specific implementation of the protocols that only the cryptographic components recommended in this document are used.

Remark 39 In the case of the ECKA-EG scheme, there is no mutual authentication. Here, only one party proves to the other to be in possession of a private key, and this also takes place only implicitly after the execution of the protocol via possession of the negotiated secret.

Appendix B.
Additional functions and algorithms

For some of the cryptographic mechanisms recommended in this Technical Guideline, additional functions and algorithms are required to, for example, generate system parameters or to generate symmetric keys from the output of random number generators or key agreement schemes. These functions and algorithms must be chosen carefully in order to achieve the security level required in this Technical Guideline and to prevent cryptanalytic attacks.

B.1. Key derivation

After key agreement, both parties hold a shared secret. In many cases, several symmetric keys, for example for encryption and data authentication, have to be derived from this secret. Moreover, the following objectives can also be achieved by using a key derivation function:

1. Binding of key material to protocol data (for example, name of the sender, name of the recipient ...) by using the protocol data in the key derivation function.
2. Derivation of session keys or keys for various distinct purposes from a master key also in purely symmetric cryptosystems.
3. Post-processing of random data with the aim of eliminating statistical biases when generating cryptographic keys.

The following method is recommended for all applications of key derivation functions:

Key derivation through extraction-then-expansion according to [78].

Table B.1.: Recommended method for key derivation

It is recommended to use one of the MACs recommended in Section 5.3 as a MAC function in the mentioned method. If none of the MAC functions mentioned there can be used, it is also possible to use HMAC-SHA1.

B.2. Generation of unpredictable initialisation vectors

As already described in Section 2.1.2, initialisation vectors for symmetric encryption schemes which use the "cipher block chaining" (CBC) mode of operation must be unpredictable. This does not mean that the initialisation vectors have to be treated confidentially, but merely that a potential adversary shall be practically unable to guess initialisation vectors used in the future . Furthermore, the adversary shall not be able to influence the choice of the initialisation vectors either.

Two methods are recommended for the generation of unpredictable initialisation vectors (with n being the block size of the block cipher used):

1. **Random initialisation vectors:** Generate a random bit sequence with a length n and use it as initialisation vector. The entropy of the random bit sequence must be at least 95 bits.
2. **Encrypted initialisation vectors:** Use a deterministic method to generate pre-initialisation vectors (e.g. a counter). Encrypt the pre-initialisation vector with the block cipher and key currently in use and use the ciphertext as an initialisation vector.

Table B.2.: Recommended methods for the generation of unpredictable initialisation vectors

For the second method, it must be ensured that the pre-initialisation vectors do not repeat themselves during the service life of the system. If a counter is used as a pre-initialisation vector, this means that counter overflows must not occur throughout the entire service life of the system.

B.3. Generation of EC system parameters

The security of asymmetric schemes based on elliptic curves rests on the assumed difficulty of the discrete logarithm problem in these groups.

In order to define the EC system parameters, the following information is required:

1. a prime number p,
2. curve parameters $a, b \in \mathbb{F}_p$ with $4a^3 + 27b^2 \neq 0$, which define an elliptic curve

$$E = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p;\ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}_E\}$$

   and
3. a base point P on $E(\mathbb{F}_p)$.

The values (p, a, b, P, q, i) then form the *EC system parameters*, where $q := \mathrm{ord}(P)$ describes the order of the base point P in $E(\mathbb{F}_p)$ and $i := \#E(\mathbb{F}_p)/q$ is the so-called *co-factor*.

Not all EC system parameters are suitable for asymmetric schemes based on elliptic curves which are recommended in this document, or in other words, the discrete logarithm problem in the groups generated by the unsuitable elliptic curves can be solved efficiently. In addition to an adequate bit length of q, the following conditions must also apply:

1. the order $q = \mathrm{ord}(P)$ of the base point P is a prime number differing from p,
2. $p^r \neq 1 \bmod q$ for all $1 \leq r \leq 10^4$, and
3. the class number of the maximal order of the endomorphism ring of E is at least 200.

For an explanation, see [38].

EC system parameters which fulfil the conditions above are also referred to as *cryptographically strong.*

Remark 40 It is recommended not to generate the system parameters on one's own, but to use standardised values instead which are provided by a trusted party.

The system parameters listed in Table B.3 are recommended:

1. brainpoolP256r1, see [38],
2. brainpoolP320r1, see [38],
3. brainpoolP384r1, see [38],
4. brainpoolP512r1, see [38].

Table B.3.: Recommended EC system parameters for asymmetric schemes which are based on elliptic curves

B.4. Generation of random numbers for probabilistic asymmetric schemes

In this Technical Guideline, several asymmetric schemes are addressed which require random numbers $k \in \{1, \ldots, q-1\}$ (e.g. as an ephemeral key), whereby q is usually not a power of 2. It has already been pointed out in the Remarks 8, 7, 19 and 21 that k should be chosen (at least nearly) uniformly distributed if possible.

The random number generators described in Chapter 9, however, generate uniformly distributed random numbers in $\{0, 1, \ldots, 2^n - 1\}$ ('random n bit strings'). The task is therefore to derive (at least nearly) uniformly distributed random numbers in $\{0, 1, \ldots, q\}$ from this.

In Table B.4, two methods are described with which this can be done. Here, $n \in \mathbb{N}$ is chosen in such a way that $2^{n-1} \leq q < 2^n - 1$ (in other words: q has the bit length n).

Method 1.

1. Choose $k \in \{0, 1, \ldots, 2^n - 1\}$ uniformly distributed.
2. `if` $k < q$ `return` k
3. `else go to` 1.

Method 2.

1. Choose $k' \in \{0, 1, \ldots, 2^{n+64} - 1\}$ uniformly distributed.
2. $k = k' \bmod q$ (i.e. $0 \leq k < q$).
3. `return` k.

Table B.4.: Computation of random values in $\{0, \ldots, q-1\}$

Remark 41 (i) Method 1 transfers a uniform distribution on $\{0, \ldots, 2^n - 1\}$ into a uniform distribution on $\{0, \ldots, q-1\}$. In fact, method 1 provides the conditional distribution on $\{0, \ldots, q-1\} \subset \{0, \ldots, 2^n - 1\}$. In contrast, method 2 does not create a (perfect) uniform distribution on $\{0, \ldots, q-1\}$, not even for ideal random number generators with values in $\{0, \ldots, 2^n - 1\}$. The deviations, however, are so small that they cannot be exploited by an adversary on the basis of the current state of knowledge.

(*ii*) The second method, however, has the advantage that any skewnesses on $\{0, \ldots, 2^n - 1\}$ should usually be reduced. Therefore, this method is recommended for random number generators in conformity with PTG.2.

(*iii*) Furthermore, method 1 has the disadvantage that the number of iterations (and thus the runtime) is not constant. For some applications, however, it can be necessary to guarantee an upper runtime limit. (It should be noted that the probability that a random number distributed uniformly in $k \in \{0, 1, \ldots, 2^n - 1\}$ is smaller than q is $q/2^n \geq 2^{n-1}/2^n = 1/2$.)

B.5. Prime generation

B.5.1. Preliminary remarks

When defining the system parameters for RSA-based asymmetric schemes, two prime numbers p, q must be chosen. For the security of these schemes, it is also necessary to keep the prime numbers secret. This requires, in particular, that p and q have to be chosen randomly. On the other hand, with regards to the usability of the application it is also important that prime generation be efficient. In the course of striving for efficiency, however, proprietary speed optimisations in key generation can cause significant cryptographic weaknesses, see e.g. [72]. As a general cautionary remark, it is therefore recommended to use mechanisms that are publicly known and which have been validated for their security.

Routines for generating random primes are also needed when system parameters for cryptosystems based on elliptic curve or finite field arithmetic without special properties are to be generated. Technical requirements on these routines, however, will differ because these prime numbers do not need to be kept secret, while on the other hand it may be important that their generation be *verifiably random*. Information on this topic can be found in section B.3.

B.5.2. Conforming methods

Three methods of prime generation are in conformance to the present Technical Guideline. Briefly summarising, these are given as follows:

1. Uniform prime generation by rejection sampling;

2. Uniform generation of an invertible residue class r with respect to $B\#$, where $B\#$ is the *primorial* of B, i.e. the product of all primes less than B, followed by the choice of a prime p of suitable size with $r \equiv p \bmod B\#$ by rejection sampling.

3. Generation of a random number s of the desired size which is coprime to $B\#$ followed by a search for the next prime number in the arithmetic sequence with step size $B\#$ starting with s.

The first two methods are equally recommended here. The third method induces certain statistical biases in the distribution of the generated primes which are generally unwanted. It is, however, a widely used method (see e.g. Table 1 of [95]) and there is no indication that these biases can be used for attack. Hence, this method is accepted as a legacy method in the present Technical Guideline.

The following tables give an overview of the three methods supported by this Technical Guideline:

Input: an interval $I := [a, b] \cap \mathbb{N}$, in which a prime is to be found.

1. Choose an odd number p according to the uniform distribution on I.
2. If p is not prime, return to step 1.
3. Output p.

Table B.5.: Recommended method 1: prime generation by rejection sampling

Input: an interval $I := [a, b] \cap \mathbb{N}$ in which a prime is to be found, as well as a small natural number B satisfying $\Pi := B\# \ll b - a$. Π may be a fixed value here if it is clear that it is much smaller than $b - a$.

1. Choose randomly according to the uniform distribution an invertible element $r \in \mathbb{Z}/(\Pi)$. This is equivalent to randomly choosing $r < \Pi$ coprime to Π.
2. Randomly choose $k \in \mathbb{N}$ subject to the constraint that $p := k\Pi + r \in I$. Hence, k is to be selected according to the uniform distribution on $[\lceil (a - r)/\Pi \rceil, \lfloor (b - r)/\Pi \rfloor]$.
3. If p is not prime, return to step 2.
4. Output p.

Table B.6.: Recommended method 2: prime generation by more efficient rejection sampling

Input: an interval $I := [a, b] \cap \mathbb{N}$ in which a prime is to be found, as well as a small natural number B satisfying $\Pi := B\# \ll b - a$. Π may be a fixed value here if it is clear that it is much smaller than $b - a$.

1. Choose randomly according to the uniform distribution an invertible element $r \in \mathbb{Z}/(\Pi)$. This is equivalent to randomly choosing $r < \Pi$ coprime to Π.
2. Randomly choose $k \in \mathbb{N}$ subject to the constraint that $p := k\Pi + r \in I$ ist. Hence, k is to be selected according to the uniform distribution on $[\lceil (a - r)/\Pi \rceil, \lfloor (b - r)/\Pi \rfloor]$.
3. Check if p is prime. If it is not, add Π and repeat step 3. If the new p is outside the interval I, return to step 1.
4. Output p.

Table B.7.: Legacy method: prime generation by incremental search

Primality testing will in the above algorithms usually be done by probabilistically, for reasons of efficiency. The following algorithm is recommended:

Miller-Rabin, see [71], algorithm 4.24.

Table B.8.: Recommended probabilistic primality test

Remark 42 (*i*) In addition to the number p to be examined, the Miller-Rabin algorithm needs a random value $a \in \{2, 3, \ldots, p-2\}$, the so-called basis. If a has been chosen in $\{2, 3, \ldots, p-2\}$ with respect to the uniform distribution, the probability that p is a composite number, although the Miller-Rabin algorithm says that p is a prime number, does not exceed $1/4$.

(*ii*) (Worst case) In order to restrict the probability that a fixed number p is identified as a prime number by means of the Miller-Rabin algorithm, although it is a composite number, to $1/2^{100}$, the algorithm must be called up 50 times with bases $a_1, \ldots, a_{50} \in \{2, 3, \ldots, p-2\}$ that were chosen independently of each other with respect to the uniform distribution. See Section B.4 for recommended methods for the calculation of uniformly distributed random numbers in $\{2, 3, \ldots, p-2\}$.

(*iii*) (Average case) In order to test with the desired certainty an odd number $p \in [2^{b-1}, 2^b - 1]$ that has been chosen with respect to the uniform distribution for primality, considerably less iterations of the Miller-Rabin algorithm than the estimate represented above would suggest are sufficient, see [35], [42], Appendix F and [55], Annex A. For example, only 4 iterations are required for $b = 1024$ in order to exclude, with a remaining error probability of 2^{-109}, that p is a composite number, although the Miller-Rabin algorithm identifies p as a prime number [55]. Here, too, the bases must be chosen independently of each other and with respect to the uniform distribution on $\{2, 3, \ldots, p-2\}$. The specific number of necessary operations depends on the bit length of p, since the density of numbers to which the estimates of the worst case scenario apply is decreasing significantly with the increasing size of the numbers.

(*iv*) (Optimisations) To optimise the runtime of, for instance, algorithm B.5 it may make sense to eliminate composite numbers with very small factors by means of trial division or sieving techniques prior to applying the probabilistic primality test. Such a preliminary test has only minor impacts on the probability that numbers classified by the test as probable prime numbers are still composite numbers. The recommendations for the required number of repetitions of the Miller-Rabin primality test therefore apply unchanged to versions optimised to this type.

(*v*) (Other comments) In accordance with [42], Appendix F.2, it is also recommended in this Technical Guideline to perform a verification of the prime number property with 50 rounds of the Miller-Rabin primality test when generating prime numbers which are to be used in particularly security-critical functions in a cryptosystem or the generation of which is not very time-critical. This applies, for example, to prime numbers which are generated once as permanent parameters of a cryptographic mechanism and are not changed over a longer period of time or possibly used by many users.

A random bit generator of functionality class PTG.3 or DRG.4 may be used to generate the required randomness. Until the end of 2022 it is also permissible to use a random bit generator of functionality class PTG.2 or DRG.3. While from an information theoretic point of view, usage of a deterministic random bit generator precludes any possibility of uniform prime generation, security is not affected: a random bit generator of functionality class DRG.3 should under cryptographic standard assumptions generate an output distribution that cannot by any known potentially practical (non-quantum) attack be distinguished from an ideal distribution.

However, it should be noted in this context that the security level of the generated RSA moduli may be limited by the security level of random bit generation. This would for instance be the case if a random bit generator with 100 bit security is used to generate RSA keys of 4096 bit length.

(*vi*) (Alternative primality tests) The choice of primality test is from a cryptanalytic point of view not security critical if the chosen test does not misclassify primes as composite and if the probability that composite numbers will pass is negligible. It is therefore possible to use other tests instead of Miller-Rabin without losing conformance to this Technical Guideline if these properties are evident from the scientific literature. However, with regards to verifying the correctness of an implementation and checking side channel resistance the use of the very widely known Miller-Rabin method is advantageous.

B.5.3. Generating prime pairs

To ensure the security of key pairs for which the underlying RSA modulus has been generated by multiplying two primes generated independently of each other with one of the approved methods, it is important that the interval $I := [a, b] \cap \mathbb{N}$ not be too small. If the modulus is supposed to have a predetermined bit length n, it is natural to take $I = [\lceil \frac{2^{(n/2)}}{\sqrt{2}} \rceil, \lfloor 2^{(n/2)} \rfloor] \cap \mathbb{N}$.

Different choices of I are in conformance to the present Technical Guideline if p and q are drawn from the same interval I and if $Card(I) \geq 2^{-8}b$.

B.5.4. Notes on the security of the recommended mechanisms

For the rest of the section, let π be the prime number function, i.e. $\pi(x) := Card(\{n \in \mathbb{N} : n \leq x, n \text{ prim } \})$. According to the prime number theorem, $\pi(x)$ is asymptotically equivalent to $x/\ln(x)$, i.e. $\frac{\ln(x)\cdot\pi(x)}{x}$ tends to one when $x \to \infty$. The security of the methods for prime generation here recommended rests on the following observations:

- All three methods are capable of generating any prime contained in the given interval if the underlying random bit generator is capable of generating all prime candidates from the interval.

- The methods B.5 and B.6 generate prime numbers the distribution of which cannot be distinguished in a practically relevant way from uniform selection of primes when the recommended security parameters are used. This is obvious for method B.5. For method B.6, this is heuristically due to the quantitative version of Dirichlet's theorem on arithmetic progressions: the relative frequency of primes among all numbers is asymptotically the same in all invertible residue classes modulo Π, and the residue class modulo Π of the prime to be generated is chosen according to the uniform distribution auf $(Z/\Pi)^*$.

- The argument just given in support of method B.6 being secure does not strictly give any guarantee that with a concrete Π and a given interval I the incidence of prime numbers is independent of the chosen residue class. Indeed, it is obvious that this will not be the case as Π approaches the order of magnitude of $b-a$. It is however expected that no significant differences among the residue classes will exist if the number of primes within each residue class is large. The interval I contains $\pi(b) - \pi(a)$ primes. Therefore, the expected number of primes per residue class mod Π is $\frac{\pi(b)-\pi(a)}{\varphi(\Pi)}$. For numbers of order 1000 bits the relative error in the approximation $\frac{b\ln(a)-a\ln(b)}{\ln(a)\ln(b)\varphi(\Pi)}$ should be very low if $\varphi(\Pi)$ is small compared to the numerator. It is recommended to choose Π in such a way that $\frac{b\ln(a)-a\ln(b)}{\ln(a)\ln(b)\varphi(\Pi)} \geq 2^{64}$.

- The qualitative reasoning given above is sufficient to assess method B.6 as suitable. More detailed investigations of closely related methods of prime generation can, however, be found in the scientific literature, see e.g. [46].

- Method B.7 generates primes that deviate from a uniform distribution. The probability of a prime $p \in I$ to be chosen by this method is proportional to the length of the preceding

prime-free stretch within the arithmetic sequence $p - k\Pi, p - (k-1)\Pi, \ldots, p - \Pi, p$ which is terminated by p. As the density of primes in these sequences tends to increase with Π it is expected that this effect is most pronounced for $\Pi = 2$. Even then, it implies only a very modest loss of entropy in practice. The bias of the induced distribution can be limited further by terminating and restarting the search for a prime if after a reasonable number k of steps no prime has been found: in this case all primes that follow a gap of length $\geq k$ will be chosen with the same probability. The biases induced by method B.7 do not appear a cause of concern as regards possible exploitation by an adversary.

Appendix C.

Protocols for special cryptographic applications

In this section, we describe some protocols which can be used as modules of cryptographic solutions. At the current time, the only protocol discussed here is SRTP, since the corresponding content on TLS, IPsec and SSH was moved to parts 2-4 of the present Technical Guideline. In general, using established protocols when developing cryptographic systems has the advantage that a comprehensive public analysis can be fallen back on. In contrast, in-house developments can easily contain vulnerabilities which the developer may only recognise with difficulty. It is therefore recommended to prefer publicly available protocols that have been evaluated several times over in-house protocol developments in cases in which this is possible.

C.1. SRTP

SRTP is a protocol which supplements the audio and video protocol RTP by functions to secure the confidentiality and integrity of the transmitted messages. It is defined in RFC 3711 [7]. SRTP must be combined with a protocol for key management, as it does not provide its own mechanisms for the negotiation of a cryptocontext.

We recommend the following use of SRTP:

- As symmetric encryption schemes, both AES in the counter mode and in the f8 mode as described in [7] are recommended.
- For the protection of integrity, an HMAC based on SHA-2 (preferably) or SHA-1 should be used. This HMAC may be reduced to 80 bits in the context of this protocol.
- As key management system, MIKEY [5] should be used. In this respect, the following key management schemes from [5] are recommended: DH key exchange with authentication via PKI, RSA with PKI, and pre-shared keys. In general, only cryptographic mechanisms recommended in this Technical Guideline should be used as components within MIKEY and SRTP.
- zRTP should only be used if it involves disproportionately high effort to solve the problem of the key distribution by means of a public-key method using a PKI or by the pre-distribution of the secret keys.
- It is urgently recommended to use the replay and integrity protection mechanisms provided in [7] in SRTP.

When using applications for the secure transmission of audio and video data in realtime, particular attention should be paid to minimising the development of side channels, for example, by means of the data transmission rate, the chronological sequence of different signals or other traffic analyses. Otherwise, attacks as in [6] become possible.

Bibliography

[1] M. Abdalla, M. Bellare, P. Rogaway, *DHIES: An encryption scheme based on the Diffie-Hellman Problem*. Available at http://charlotte.ucsd.edu/~mihir/papers/dhaes.pdf

[2] AIS 20: *Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren* [Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators], Version 3, 15.5.2013 Federal Office for Information Security (BSI). Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_pdf.pdf.

[3] AIS 31: *Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren* [Functionality Classes and Evaluation Methodology for Physical Random Number Generators], Version 3, 15.5.2013 Federal Office for Information Security (BSI). Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_pdf.pdf.

[4] W. Killmann, T. Lange, M. Lochter, W. Thumser, G. Wicke, Annex to AIS 46: *Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations* Federal Office for Information Security (BSI). Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_ECCGuide_e_pdf.pdf.

[5] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, *MIKEY: Multimedia Internet KEYing*, RFC 3830. Available at http://tools.ietf.org/html/rfc3830.

[6] Ballard, Coulls, Monrose, Masson, *Spot me if you can: recovering spoken phrases in encrypted VoIP conversations*, IEEE Symposium on Security and Privacy, 2008.

[7] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman *The Secure Real-time Transport Protocol (SRTP)* RFC 3711, 2004.

[8] M. Bellare, R. Canetti und H. Krawczyk. *HMAC: Keyed-Hashing for Message Authentication.* RFC 2104, 1997.

[9] M. Bellare, K. Miner, *A Forward-Secure Digital Signature Scheme* Advances in Cryptology – Crypto 1999, LNCS 1666/1999, 431-448.

[10] BSI, *Entwicklungsstand Quantencomputer*, https://www.bsi.bund.de/qcstudie, 2018

[11] D. Gillmor, *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*, RFC 7919, 2016

[12] D. Bernstein, *Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?*, SHARCS 2009.

[13] A. Biryukov und D. Khovratovich, *Related-Key Cryptanalysis of the Full AES-192 and AES-256*, Asiacrypt 2009, LNCS 5912/2009, 1-18.

[14] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, A. Shamir, *Key Recovery Attacks of Practical Complexity on AES Variants With Up to 10 Rounds*, Eurocrypt 2010, LNCS 6110/2010, 299-319.

[15] S. Blake-Wilson, A. Menezes, *Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol*, Public Key Cryptography 1999, LNCS 1560/1999, 154-170.

[16] D. Bleichenbacher (1998). *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1.* In Advances in Cryptology—CRYPTO'98 (S. 1-12). Springer Berlin/Heidelberg.

[17] BNetzA, Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen) [Publication on the electronic signature according to the German Signature Act [SigG] and the German Signature Ordinance [SigV] (Overview of suitable algorithms)], published on the websites of the Federal Gazette [German Bundesanzeiger] under BAnz AT 27.03.2013 B4.

[18] A. Bogdanov, D. Khovratovich, C. Rechberger, *Biclique Cryptanalysis of the Full AES*, Asiacrypt 2011, LNCS 7073/2011, 344-371.

[19] H. Böck, J. Somorovsky, C. Young, *Return Of Bleichenbacher's Oracle Threat (ROBOT)*, IACR e-print report 1189/2017

[20] D. Boneh und G. Durfee. *Cryptanalysis of RSA with private key d less then* $N^{0.292}$. Eurocrypt 1999, LNCS 1592/1999, 1-11.

[21] D. Brown, *Generic Groups, Collision Resistance, and ECDSA*, Designs, Codes and Cryptography 04/2005, Vol. 35, Issue 1, pp. 119-152.

[22] T. Brown, R. Gallant, *The Static Diffie-Hellman Problem*, IACR e-print report 306/2004.

[23] BSI, TR 02102-2, *Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 - Verwendung von Transport Layer Security (TLS)* [Cryptographic Mechanisms: Recommendations and Key Lengths, Part 2 - Use of Transport Layer Security (TLS)], current version available at https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html.

[24] BSI, TR 02102-3, *Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 3 - Verwendung von IPsec* [Cryptographic Mechanisms: Recommendations and Key Lengths, Part 3 - Use of IPsec], current version available at https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html.

[25] BSI, TR 02102-4, *Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 4 - Verwendung von Secure Shell (SSH)* [Cryptographic Mechanisms: Recommendations and Key Lengths, Part 4 - Use of Secure Shell (SSH)], current version available at https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html.

[26] BSI, TR 3110-2, *Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token*, Version 2.20, 3.2.2015, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-2-V2_2.pdf.

[27] BSI, *Elliptic Curve Cryptography*, BSI Technical Guideline TR-03111, Version 2.0, 2012, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_pdf.html.

[28] BSI, Technical Guideline TR-03116, *Technische Richtlinie für eCard-Projekte der Bundesregierung* [Technical Guideline for eCard Projects of the Federal Government], current version available at https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03116/index_htm.html.

[29] BSI, Technical Guideline TR-03116-2, *eCard-Projekte der Bundesregierung, Teil 2 – Hoheitliche Ausweisdokumente* [eCard Projects of the Federal Government, Part 2 – Official Identification Documents], current version available at https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03116/index_htm.html.

[30] BSI, BSI Technical Guideline TR-03125, *Beweiswerterhaltung kryptographisch signierter Dokumente* [Preservation of Evidence of Cryptographically Signed Documents], 2011, https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03125/index_htm.html.

[31] J. Buchmann, E. Dahmen, A. Hülsing *XMSS–A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions*, Fourth International Conference in Post-Quantum Cryptography, LNCS 7071/2011, 117-129.

[32] S. Chen, R. Wang, X. Wang, K. Zhang, *Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow*, 2010 IEEE Symposium on Security and Privacy. Available at http://research.microsoft.com/pubs/119060/WebAppSideChannel-final.pdf.

[33] J. H. Cheon, *Security Analysis of the Strong Diffie-Hellman Problem*, Advances in Cryptology - Eurocrypt 2006, LNCS 4004/2006, pp. 1-11.

[34] J.-S. Coron, D. Naccache and J. Stern. *On the Security of RSA-Padding.* Crypto 99, LNCS 1666/1999, 1-18.

[35] I. Damgard, P. Landrock und C. Pommerance. *Average Case Error Estimates for the Strong Provable Prime Test*, Mathematics of Computation, Vol. 61, No. 203, 177-194, 1993.

[36] M. Daum and S. Lucks. *The Story of Alice and Bob,* rump session presentation, Eurocrypt 2005, http://www.cits.rub.de/imperia/md/content/magnus/rumpec05.pdf.

[37] W. Diffie, P. van Oorschot, M. Wiener, *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, 1992 Vol. 2, Number 2, pp. 107-125.

[38] M. Lochter, J. Merkle, RFC 5639: *Elliptic Curve Cryptography ECC Brainpool Standard Curves and Curve Generation*, 2010. Available at http://tools.ietf.org/html/rfc5639.

[39] ECRYPT II, *Yearly Report on Algorithms and Key Sizes*, 2011.

[40] Federal Information Processing Standards Publication 140-2 (FIPS PUB 140-2), *Security Requirements for Cryptographic Modules*, 2002.

[41] Federal Information Processing Standards Publication 180-4 (FIPS PUB 180-4), *Secure Hash Standard*, 2012.

[42] Federal Information Processing Standards Publication 186-4 (FIPS PUB 186-4), *Digital Signature Standard (DSS)*, 2013.

[43] Federal Information Processing Standards Publication 197 (FIPS PUB 197), *Advanced Encryption Standard (AES)*, 2001.

[44] Federal Information Processing Standards Publication 202 (FIPS PUB 202), *SHA-3 Standard: Permutation-Based Hash and Extendable Output Functions*, http://nvlpubs.nist.gov/nistpubs/fips/NIST.FIPS.202.pdf, 2015.

[45] N. Ferguson, *Authentication Weaknesses in GCM*, 2005. Available at http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf.

[46] P.A. Fouque, M. Tibouchi (2014), *Close to uniform prime number generation with fewer random bits*, In International Colloquium on Automata, Languages, and Programming (pp. 991-1002). Springer, Berlin, Heidelberg.

[47] M. Gebhardt, G. Illies, W. Schindler. *A Note on the Practical Value of Single Hash Collisions for Special File Formats,* Sicherheit 2006, K"ollen-Verlag, LNI P-77 (2006), 333-344. Extended version: NIST Cryptographic Hash Workshop 2005. Available at http://csrc.nist.gov/groups/ST/hash/documents/Illies_NIST_05.pdf.

[48] N. Heninger, Z. Durumeric, E. Wustrow, J. Halderman, *Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices*, Proceedings of the 21st Usenix Symposium, 08/2012. Available at https://www.factorable.net/weakkeys12.conference.pdf.

[49] G. Illies, M. Lochter, O. Stein *Behördliche Vorgaben zu kryptographischen Algorithmen*, Datenschutz und Datensicherheit 11/2011, pp. 807-811.

[50] IEEE P1363. *Standard Specifications for Public Key Cryptography*, 2000.

[51] ISO/IEC 11770-2-2008 *Information technology – Security techniques – Key management - Part 2: Mechanisms using symmetric techniques*, 2008.

[52] ISO/IEC 11770-3-2008. *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*, 2008.

[53] ISO/IEC 14888-2-2008 *Information technology – Security techniques – Digital signatures with appendix – Part 2: Integer factorization based mechanisms*, 2008.

[54] ISO/IEC 14888-3-2006. *Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*, 2006.

[55] ISO/IEC 18032. *IT security techniques – Prime number generation,* 2005.

[56] ISO/IEC 9796-2-2010. *Information technology – Security techniques – Digital Signature Schemes giving message recovery – Part 2: Integer Factorization based mechanisms*, 2010.

[57] ISO/IEC 9797-1-2011. *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*, 2011.

[58] G. Itkis, *Forward Security, adaptive cryptography: Time Evolution*, 2004. Available at http://www.cs.bu.edu/~itkis/pap/forward-secure-survey.pdf.

[59] T. Iwata, K. Ohashi, K. Minematsu, *Breaking and Repairing GCM Security Proofs*, Crypto 2012, LNCS 7417/2012, pp. 31-49.

[60] J. Kelsey, B. Schneier, D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Crypto 96, LNCS 1109/1996, pp. 237-251.

[61] W. Killmann, W. Schindler: *A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators*, Version 3.1, 25.09.2001. Federal Office of Information Security (BSI), previous mathematical-technical annex to [3].[1] Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_evaluation_methodology_for_true_RNG_e.pdf.

[1] Still relevant to aspects which are not covered by other annexes to [3].

[62] W. Killmann, W. Schindler: *Functionality classes for random number generators.* Federal Office of Information Security (BSI), Version 2.0, 18.09.2011, mathematical-technical annex to [2] and [3]. Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf.

[63] A.K. Lenstra und E.R. Verheul. *Selecting Cryptographic Key Sizes.* J. Cryptology 39, 2001. 255-293.

[64] A.K. Lenstra. *Key lengths*, in: *Handbook of Information Security*, John Wiley & Sons, 2006, 617-635.

[65] S. Lucks, *Attacking Triple Encryption*, Fast Software Encryption 1998, LNCS Vol. 1372, pp. 239-253.

[66] ISO/IEC 18033-2. *Information Technology – Security techniques – Part 2: Asymmetric Ciphers*, 2006.

[67] V. Gayoso Martínez, F. Hernández Álvarez, L. Hernández Encinas, C. Sánchez Ávila, *A Comparison of the Standardized Versions of ECIES*, Sixth International Conference on Information Assurance and Security, 2010.

[68] R. Merkle, *Secure Communications over Insecure Channels*, Communications of the ACM, Vol. 21, No. 4, 1978, pp. 294-299.

[69] R. Merkle, M. Hellman, *On the security of multiple encryption*, Communications of the ACM, Vol. 24, No. 7, 1981, pp. 465-467.

[70] P. van Oorschot, M. Wiener, *A known-plaintext attack on two-key triple encryption*, Eurocrypt 1990, LNCS Vol. 473, pp. 318-323.

[71] A. Menezes, P. van Oorschot und O. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[72] M. Nemec, M. Sys, P. Svenda, D. Klinec, V. Matyas, (2017, October). *The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli.* In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1631-1648). ACM.

[73] NIST. Recommendation for Block Cipher Modes of Operation: Methods and Techniques, Special Publication SP800-38A, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2001.

[74] NIST Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, Special Publication SP800-38E, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2010.

[75] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication SP800-38B, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 2005.

[76] NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication, Special Publication SP800-38D, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, November 2007.

[77] NIST Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, Special Publication SP800-56B Rev. 1, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 10/2014.

[78] NIST Recommendation for Key Derivation through Extraction-then-Expansion, Special Publication SP800-56C, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 11/2011.

[79] NIST Electronic Authentication Guideline, Special Publication SP800-63-2, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 08/2013.

[80] NIST Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication SP800-67 Rev.1, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 01/2012.

[81] NIST Recommendation for Key Derivation Using Pseudorandom Functions, Special Publication SP800-108, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, 10/2009.

[82] P. Nguyen and I. Shparlinski. The insecurity of the elliptic curve signature algorithm with partially known nonces. *Designs, Codes and Cryptography*, Vol. 30, Number 2, 201-217, 2003.

[83] J.F. Raymond, A. Stiglic, *Security Issues in the Diffie-Hellman Key Agreement Protocol*, IEEE Transactions on Information Theory, 2000.

[84] T.Kivinen, M. Kojo More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) *RFC 3526*, May 2003

[85] S. Kent. IP Encapsulating Security Payload (ESP). *RFC 4303*, 2005.

[86] R. Housley. Cryptographic Message Syntax (CMS). *RFC 5652,* 2009.

[87] T. Ristenpart, S. Yilek, *When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography* Proceedings of the Network and Distributed Systems Security Symposium - NDSS 2010, 2010.

[88] PKCS #1 v2.2: RSA Cryptographic Standard, 27.10.2012. Available at https://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf.

[89] W. Schindler: *Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren* [Functionality Classes and Evaluation Methodolody for Deterministic Random Number Generators]. Federal Office of Information Security, Version 2.0, 02.12.1999, previous mathematical-technical annex to [2].[2] Available at https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_Functionality_Classes_Evaluation_Methodology_DRNG_e.pdf.

[90] A. Shamir. *How to share a secret.* Communications of the ACM, Vol. 22 Issue 11 (1979), 612-613.

[91] SOGIS, *SOGIS Agreed Cryptographic Mechanisms* Version 1.0, Mai 2016, URL: http://www.sogisportal.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.0.pdf

[2] Still relevant to aspects which are not covered by other annexes to [2].

[92] D. X. Song, D. Wagner, X. Tian, *Timing Analysis of Keystrokes and Timing Attacks on SSH*, Proceedings of the 10th USENIX Security Symposium, 2001.

[93] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, Y. Markov, (2017). *The first collision for full SHA-1.* IACR Cryptology ePrint Archive, 2017, 190.

[94] M. Stevens, *Attacks on Hash Functions and Applications*, dissertation, Leiden University, 2012.

[95] P. Švenda, M. Nemec, P. Sekan, R. Kvašňovský, D. Formánek, D. Komárek, V. Matyáš (2016), *The Million-Key Question–Investigating the Origins of RSA Public Keys*, In: 25th USENIX Security Symposium. Proceedings.

[96] S. Vaudenay, *Security Flaws Induced by CBC Padding: Applications to SSL, IPSEC, WTLS...*, Eurocrypt 2002, LNCS 2332/2002, 534-545.

[97] X. Wang, Y.L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. Crypto 2005, LNCS 3621/2005, 17-36.

[98] atsec, *Dokumentation und Analyse des Linux-Pseudozufallszahlengenerators* [Documentation and Analysis of the Linux Pseudo-Random Number Generator], study on behalf of the BSI, 2013, https://www.bsi.bund.de/DE/Publikationen/Studien/LinuxRNG/index_htm.html.